Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Системы автоматики, автоматизированное управление и проектирование

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Ченцов С.В.
«19» 06 2017 г.

# МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

# ГЕНЕРАТОР КОММУТАЦИОННЫХ СХЕМ

09.04.02 Информационные системы и технологии

09.04.02.02 Информационные системы и технологии в управлении технологическими процессами

| | | | |
|---|---|---|---|
| Научный руководитель | _____ | 16.06.2017г. | доцент, канд. техн. наук Е.Е. Носкова |
| Выпускник | _____ | 16.06.2017г. | А.О. Щербаков |
| Рецензент | _____ | 16.06.2017г. | доцент, канд. техн. наук Е.П. Моргунов |
| Нормоконтролер | _____ | 16.06.2017г. | Т.А. Грудинова |

Красноярск 2017

# РЕФЕРАТ

Выпускная квалификационная работа по теме «Генератор коммутационных схем» содержит 108 страниц текстового документа, 18 использованных источников, 36 иллюстрации и 2 таблицы.

КОММУТАЦИОННАЯ СХЕМА, ЭЛЕКТРОННЫЙ КОМПОНЕНТ, КОЛИЧЕСТВО СВЯЗЕЙ, КОЛИЧЕСТВО ЭЛЕМЕНТОВ, МНОГОЭЛЕМЕНТНАЯ СХЕМА, МНОГОСВЯЗНАЯ СХЕМА, АЛГОРИТМ РАЗМЕЩЕНИЯ, ПЕЧАТНАЯ ПЛАТА, ПРОГРАММА- ГЕНЕРАТОР СХЕМ.

**Актуальность.** Актуальность диссертационной работы заключается в повышении степени автоматизации процесса размещения за счёт получения информации о качестве работы алгоритмов путём генерации данных для тестирования алгоритмов размещения.

**Цель.** разработка механизма создания коммутационных схем с заданной элементной базой для получения тестовых данных для оценки качества решения задачи размещения.

**Задачи:**

— Анализ и выявление параметров генерации;

— Создание алгоритма формирования исходных данных для генерации;

— Создание алгоритмов генерации;

— Создание программы-генератора.

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Современные системы проектирования печатных плат не являются полностью автоматизированными. Участие человека в процессе изготовление макета печатной платы до сих пор обязательно. Задача размещения электронных компонентов на плоскости печатной платы является одной из основных задач в процессе проектирования электронного устройства.

Выбор алгоритма решения задачи размещения может существенно повлиять на качество изготовления печатной платы и работы устройства в целом. Для успешного выбора алгоритма необходима информация о эффективности его работы при проектировании коммутационных схем на заданной элементной базе. Перебор всех алгоритмов с целью поиска лучшего результата не всегда является возможным решением проблемы выбора алгоритма размещения, так как при конструировании схем с большим количеством элементов (1000 и более) алгоритм может выполнятся сутки, что существенно может отразится на сроках проектирования и, с учётом итерационности процесса проектирования, как следствие, на конкурентоспособности компании производящей печатные платы.

**Актуальность.** Актуальность диссертационной работы заключается в повышении степени автоматизации процесса размещения за счёт получения информации о качестве работы алгоритмов путём генерации данных для тестирования алгоритмов размещения.

**Цель.** разработка механизма создания коммутационных схем с заданной элементной базой для получения тестовых данных для оценки качества решения задачи размещения.

**Задачи:**

— Анализ и выявление параметров генерации;

— Создание алгоритма формирования исходных данных для генерации;

— Создание алгоритмов генерации;

— Создание программы-генератора.

## 1 Постановка задачи размещения и методы решения

Конструирование представляет собой часть процесса проектирования, цель которого состоит в разработке конструкторской документации для изготовления изделий на производственном предприятии. Этот этап начинается с передачи принципиальных схем электронных устройств, разработанных на этапе функционального проектирования с помощью CAE – части интегрированных САПР. Эта передача осуществляется довольно просто в рамках единой информационной модели ТО при использовании сквозного проектирования на основе интегрированных САПР.

Математическими моделями описания конструкций ЭУ являются структурные топологические (*2D*) и геометрические (*3D*) модели, которые образуют вместе с методами конструирования математическое обеспечение CAD-систем в рамках интегрированных САПР.

Разработка конструкций ТО любой физической природы в целом связана, прежде всего, с разработкой математического описания в виде геометрических *3D* моделей на основе методов геометрического моделирования, что выходит за рамки рассмотрения данного курса.

Алгоритмы и методы конструирования являются едиными при рассмотрении описания ТО независимо от вида структурной модели (2D или 3D).

Этап конструкторского проектирования электронных устройств определяется комплексом задач связанных с преобразованием функциональных логических или принципиальных электрических схем в совокупность конструктивных узлов, между которыми устанавливаются необходимые пространственные, механические и электрические связи. Принципом конструирования ЭУ является применение функционально-узлового (модульного) метода проектирования, предусматривающего выделение

конструктивных узлов (модулей) различной степени сложности, находящихся в отношении соподчинённости.

Таким образом, конструкция электронного устройства представляет собой иерархическую структуру, в которой узлы низшего уровня объединяются в узлы высшего уровня. Широкому практическому распространению модульного принципа конструирования содействовали такие факторы, как развитие методов микроминиатюризации электронных компонентов, создание и использование интегральных микросхем, печатных и многослойных печатных плат.

Используются следующие названия конструктивов ЭУ.

*Элемент* – выделяемый (проверяемый по своим техническим условиям) конструктив, поставляемый как отдельное изделие, например интегральная микросхема (ИС), гибридная интегральная схема (ГИС), транзистор, диод, резистор и другие электроэлементы.

*Ячейка* или *типовой элемент замены* (ТЭЗ) – это объединение элементов в едином конструктивном исполнении на основе печатной платы, в виде микроэлектронного узла или микросборки. Обычно ячейка реализует сложную функциональную схему, в большинстве случаев функционально-законченную, проверяемую по своим техническим условиям, и является съемной при ремонте и устранении неисправностей. Соединения между элементами в ячейке, как правило, печатные.

Следующие иерархические уровни занимают конструктивы, называемые *устройствами* (панелями, субблоками), *блоками* (стойками). Каждый из конструктивов последующего уровня состоит из нескольких конструктивов предыдущего уровня, размещаемых в некоторой несущей конструкции (каркасе) и соединяемых между собой проводным монтажом, гибкими шлейфами или многослойными печатными объединительными платами.

В конструкторском проектировании выделяют три *группы задач:*

—синтез конструкций;

—контроль полученных решений;

—оформление документации конструкторской (КД) и технологической (ТД), включая выпуск носителей информации.

Основными задачами синтеза конструкций ЭУ являются следующие коммутационно-монтажные задачи:

—компоновка конструктивов *i*-го уровня в конструктив *(i – 1)*-го уровня (компоновка простейших функциональных элементов логических схем И, ИЛИ, НЕ в корпуса микросхем, компоновка интегральных микросхем (ИМС) и других электрорадиоэлементах (ЭРЭ) в ячейки, ячеек в блоки и устройства);

—размещение конструктивов *i*-го в конструктивах *(i – 1)*-го уровня (размещение цифровых и аналоговых бескорпусных ИМС по подложке гибридной интегральной схемы (ГИС), размещение ЭРЭ на печатной плате, размещение ячеек в блоке и т. д.);

— трассировка монтажных соединений между конструктивами на всех уровнях (соединения между кристаллами ИМС, печатный монтаж, межплатные соединения).

На сегодняшний день существующие CAD системы предназначенные для автоматизации проектирования электроники не являются полностью автоматизированными. Это проявляется в отсутствии методов решения задачи размещения конструктивов, которые не уступают в надёжности и эффективности перед ручными способами размещения. Популярные программные продукты: OrCAD [4], Xpedition [5], Delta Design [6], Altium Designer [7] не содержат полностью автоматизированных решений выполняющие размещение конструктивов.

### 1.1 Математическое описание коммутационных схем

Любая логическая или принципиальная электрическая схема устройства ЭУ состоит из некоторого набора базовых элементов, определенным образом связанных между собой. Элементами таких схем могут быть резисторы, конденсаторы, транзисторы, логические элементы, микросхемы и т. д. Связи в схеме соответствуют подаче электрических сигналов (дискретного или аналогового типа) на выходы (или полюса) определенных элементов.

Для описания схем удобно воспользоваться символикой теории множеств. Множество - это любая определённая совокупность объектов. Объекты, из которых состоит множество, называются его элементами. Множества, как объекты могут быть элементами других множеств. Множество, элементами которого являются множества, обычно называются классом или семейством. Множество, не содержащее элементов, называется пустым.

Учитывая характер основных задач конструирования устройств ЭУ, можно рассматривать исходную схему как некоторое множество элементов $E = \{e_1, e_2, \dots, e_n\}$ соединенных между собой электрическими цепями из множества $V = \{v_1, v_2, \dots, v_m\}$. Такое представление называется схемой соединений или коммутационной схемой.

Каждый элемент схемы имеет некоторое множество соединительных выводов $E = \{e_1, e_2, \dots, e_k\}$. Кроме выводов элементов, в схеме присутствуют внешние выводы $C_0$, осуществляющие связь схемы с другими устройствами. Для дальнейшего удобно считать эти выводы принадлежащими фиктивному элементу $e_0$.

Два вывода схемы будем считать связанными, если они объединяются одной электрической цепью. Электрическая цепь в общем случае связывает больше двух выводов в схеме. Назовем комплексом совокупность эквипотенциальных выводов схемы, а число выводов в комплексе размером комплекса или размером соответствующей цепи.

Два элемента схемы считаются связанными, если имеется, по крайней мере одна электрическая цепь, содержащая выводы этих элементов. Некоторое множество элементов называется несвязанным, если любые два элемента этого множества являются несвязанными. Несвязанное множество элементов является максимальным, если добавление к нему любого другого элемента схемы нарушает свойство несвязанности элементов.

Под элементным комплексом $v'_j$ будем понимать подмножество элементов из $E=\{e_0, e, \ldots, e_n\}$, соединенных цепью $j$ $(j=1, 2, \ldots, M)$. Если каждый элемент схемы связан, по крайней мере с одним другим элементом, что естественно для реальных схем, то $E = \bigcup_{j=1}^{M} v'_j$ . Однако элементные комплексы могут содержать общие элементы, т. е. $v'_i \cap v'_j \neq 0$.

Число элементов в комплексе $v'_j$ назовем размером элементного комплекса: $p'_j=|v'_j|$, $p'_j>=1$. Случай $p'_j=1$ означает, что цепь $j$ соединяет выводы одного и того же элемента.

Характерной особенностью коммутационных схем, вызывающей в ряде случаев недоразумения при их описании, является присутствие в них цепей, соединяющих несколько элементов (выводов). Собственно схема в общем случае не задает конкретного способа реализации таких многоконцевых соединений: он должен быть определен в процессе решения основных задач конструирования.

Среди различных вариантов описания коммутационных схем наибольшей общностью и наглядностью обладает описание схемы в виде графа. Такое представление широко используется при математической постановке различных оптимизационных задач конструирования и позволяет в целом ряде случаев найти адекватные задачи в теории графов и воспользоваться при разработке алгоритмов решения задач конструирования известными математическими методами.

При выборе способа описания коммутационной схемы графом необходимо учитывать, особенности схемы, а также допустимую степень идеализации модели для использования в конкретной задаче конструирования:

при компоновке, размещении и трассировке. Рассмотрим несколько способов описания схем графами.

Первый из них является наиболее общим и предполагает построение графа коммутационной схемы (ГКС). В отличие от обычного линейного графа, определяемого заданием ребер между определенными парами вершин, в ГКС различаются несколько типов ребер и вершин, рисунок 1.1.

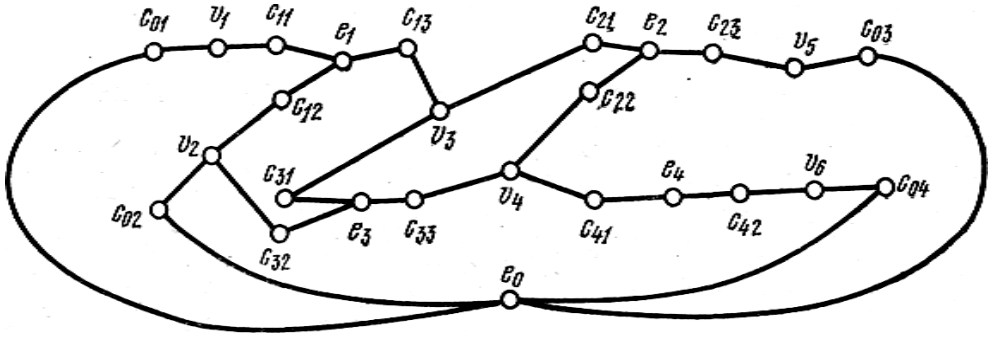



Рисунок 1.1 - Граф коммутационной схемы

Введем вершины трех типов: *E, C, V*. Вершины *E* соответствуют элементам схемы, вершины *C* – выводам элементов, включая внешние выводы схемы, а вершины *V* – цепям (комплексам) схемы. Среди ребер ГКС будем различать элементные ребра *F* и сигнальные ребра *W*.

Элементные ребра определяют принадлежность выводов из множества *C* элементам из множества *E* и задаются парами вершин $(e_i\ c_к)$.Сигнальные ребра *W* определяют вхождение выводов из *C* в отдельные цепи и описываются парами вершин$(v_i\ c_к)$.

При решении некоторых задач конструирования может оказаться необходимым дополнить введенную модель схемы. Так, иногда существенна информация об источниках и приемниках сигналов в каждой цепи, поэтому сигнальным ребрам придается соответствующая ориентация, и граф схемы становится ориентированным. В других ситуациях, например, при трассировке соединений на плоскости, необходимо учитывать порядок расположения конструктивных выводов на элементах. В связи с этим в ГКС следует уточнить порядок следования элементных ребер, что может быть достигнуто специальными средствами. В большинстве же случаев степень детализации описания коммутационной схемы, обеспечиваемая ГКС, вполне достаточна.

Известно, что произвольный неориентированный граф $G=(X,U)$ с множеством вершин $X$ и множеством ребер $U$ может быть задан в числовой форме матрицей инциденций $C=||C_i||_{n\times m}$ , в которой элемент $c_{ij}=1$, если вершина $X_i$ инцидентна ребру $u_j$, и $c_{ij}=0$ в противном случае. Учитывая, что ГКС содержит вершины и ребра разных типов, его структуру удобнее описать с помощью пары матриц $A$ и $B$.

Матрица $A$ представляет цепи схемы и определяется следующим образом: $A=||a_{ij}||_{M\times K}$ , где М – число цепей, $K$ – число выводов в схеме; элемент $a_{ij}=1$, если вывод $C$ принадлежит цепи $v_j$, и $a_{ij}=0$ в противном случае.

Матрица $B=||b_{ij}||_{n\times K}$ выделяет подмножества выводов, принадлежащие отдельным элементам. Строки матрицы соответствуют элементам, а столбцы - выводам. Элемент $b_{ij}=1$, если вывод $c_j$ принадлежит элементу, и равен нулю в противном случае.

Модель в виде ГКС используется при задании полной информации о схеме в автоматизированном процессе конструирования. Вместе с тем, при алгоритмическом решении отдельных задач конструирования удобнее пользоваться упрощенными моделями схем. Так, например, при компоновке узлов можно отождествить наборы выводов $c_i$ с самими элементами $e_i$. В результате этого преобразования комплексы $V_j$ переходят в элементные комплексы $v'_j$, что соответствует в ГКС «стягиванию» определенных подмножеств вершин из $C$ в вершины из $E$ и устранению элементных ребер $F$. Таким образом, получаем граф $G'=(E, V, W)$, подмножества вершин которого $E$ *и* $V'$ соответствуют элементам и элементным комплексам схемы, а множество ребер W определяет вхождение элементов в комплексы. Граф $G'$ является двудольным графом, поскольку и $E$, и $V$ являются несвязанными множествами вершин. Полученную модель схемы будем называть графом элементных комплексов (ГЭК), рисунок 1.2.

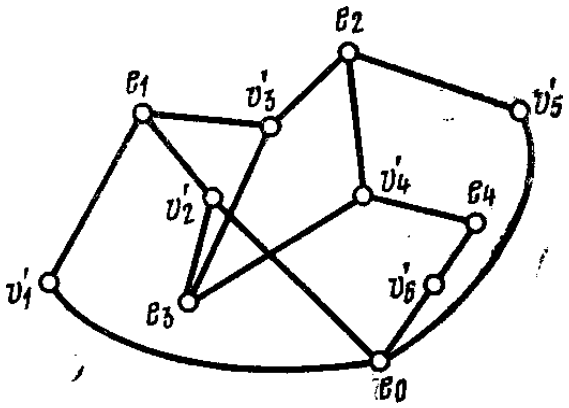


Рисунок 1.2 - Граф элементных комплексов

Для описания ГЭК удобно воспользоваться матрицей комплексов $Q$ строки которой, соответствуют элементам $e_i$, а столбцы – элементным комплексам $v'_j$. Значение $q_{ij}=1$, если элемент $e_i$ входит в комплекс $v'_j$ (связан с $j$-й цепью), и $q_{ij}=0$ в противном случае. Число единиц в любой строке матрицы равно числу цепей, связанных с соответствующим элементом. Число единиц в столбце равно размеру данного комплекса. Отметим, что одноэлементные комплексы ($p'_j=1$) приводят к образованию петель на некоторых вершинах $e_i$ в ГЭК. Такие комплексы соответствуют целям, связывающим выводы одного и того же элемента, и не оказывают влияния на решение задач компоновки и размещения. В связи с этим при задании схемы в виде ГЭК они могут быть опущены.

Заметим, что между матрицей $Q$ и введенными ранее для описания ГКС матрицами $A$ и $B$ существует простая связь:

$$Q = B \cdot A', \qquad (1.1)$$

где $A'$–транспонированная матрица $A$, символ означает умножение матриц с булевыми переменными $0$ и $1$.

Модели схемы в виде ГКС или ГЭК задают электрически связанные подмножества выводов (элементов), но оставляют свободу в определении конкретных монтажных соединений группы выводов должно быть связным графом, множество вершин которого включает данную группу выводов. Напомним, что в понятии графа существен только факт присутствия ребер между отдельными вершинами, а не способ их изображения. Так, одному и тому же

графу могут соответствовать различные чертежи, отличающиеся расположением вершин и геометрической формой соединяющих эти вершины ребер.

Расчет оптимальных конфигураций соединений составляет основу для разработки схем проводного и печатного монтажа. Графы соединений, как правило, не содержат циклов, поскольку удаление любого ребра, входящего в цикл, не нарушает электрического соединения рассматриваемой группы выводов. Поэтому можно считать, что граф, соответствующий соединению группы выводов, есть связный граф без циклов, т. е. дерево.

При проводном монтаже возможна реализация лишь элементарных соединений $(c_к, c_i)$ типа «вывод – вывод». Таким образом, дерево соединений $T=(X, U)$ для цепи размером $p$ состоит из набора вершин $x \in X$, являющихся выводами данной цепи ($|X|=p$),и множества ребер $u \in U$, соответствующих элементарным соединениям $(c_к, c_l)$, причем $|U|=p-1$.

Согласно теореме Кэли, для $p$ вершин существует $p^{p-2}$ различных деревьев.

При печатном (пленочном) способе выполнения соединений для осуществления необходимых электрических связей, помимо выводов, можно использовать внутренние точки проводников. В этом случае возможно образование соединений типа «вывод – проводник» и «проводник – проводник» что значительно увеличивает общее количество возможных деревьев соединений. Теперь дерево соединений $T_P=(X_P, U)$, помимо исходного набора вершин $X$, имеет некоторое множество дополнительных вершин: $X_P=XUP$, а множество $U$ состоит из $p+|P| - 1$ ребер. Такое дерево называется деревом Штейнера.

Учет особенностей реализации соединений позволяет использовать при решении отдельных задач компоновки и размещения упрощенные модели описания схем, основанные на задании «степени связанности» элементов друг с другом.

Один из способов состоит в следующем. Подсчитаем для каждой пары элементов число связывающих их цепей. Далее построим граф G=$(E, U)$, в котором вершины $e_i$ соответствуют элементам, а ребра $u_{ij}$ с приписанными

к ним весами $r_{ij}>0$ – количеству цепей между элементами $e_i$ и $e_j$. Полученный граф называется взвешенным графом схемы (ВГС), рисунок 1.3.

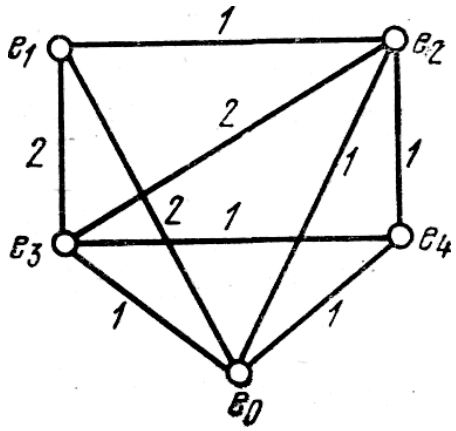



Рисунок 1.3 - Взвешенный граф схемы

Данный способ построения ВГС равносилен выполнению следующего преобразования в ГЭК схемы. Построим для каждого элементного комплекса полный граф элементарных соединений. Очевидно, что для комплекса размером $p$ получим $p(p–1)/2$ соединений. Далее, для каждой пары вершин $e_i$ и $e_j$. введем ребро, если между ними имеется хотя бы одно элементарное соединение. Припишем ребру $u_{ij}$ вес $r_{ij}$, равный числу элементарных соединений между вершинами $e_i$ и $e_j$.. Поскольку в данном случае веса $r_{ij}$ целочисленные, построенный ВГС можно считать мультиграфом, в котором веса задают кратности ребер. Результат применения указанного преобразования к графу рисунка 1.2 показан на рисунке 1.3.

В общем случае ВГС может быть описан матрицей соединений $R=||r_{ij}||_{n\times n}$ строки и столбцы которой соответствуют элементам схемы, а $r_{ij}$ равен весу, приписанному соединению элементов $e_i$ и $e_j$. Матрица $R$ – симметричная с нулевой главной диагональю ($r_{ij}=0$, i =1, 2, ..., n).

Иногда удобно считать $r_{ij} = \sum_j^n r_{ij}$ (i=1,2, …, n) равным суммарному числу цепей, связанных с элементом ei. При этом условии и рассмотренном способе расчета весов rij между матрицей соединений R и матрицей комплексов Q существует простая связь:

$$R = Q \times Q',$$

где $Q'$ – транспонированная матрица $Q$.

$$r_{ij} = \sum_{s=1}^{M} q_{is} \cdot q'_{sj} = \sum_{s=1}^{M} q_{is} \cdot q_{js}\ , \qquad (1.2)$$

Отсюда следует, что $r_{ij}$ равно числу цепей, связывающих элементы $e_i$ и $e_j$.

При $i=j$ из (4.2) получим

$$r_{ij} = \sum_{s=1}^{M} q_{is} \cdot q'_{sj} = \sum_{s=1}^{M} q_{is} \cdot q_{js} \qquad (1.3)$$

т. е. $r_{ij}$ равно количеству цепей, связывающих элемент $e_i$ с другими элементами.

Поскольку для соединения выводов одной цепи в действительности используются деревья соединений, описанный способ построения ВГС является относительно грубым. Поэтому в ряде алгоритмов размещения и компоновки используются весовые оценки, учитывающие приоритеты цепей и вероятности реализации отдельных соединений.

Широкий класс весовых оценок, используемых при образовании матрицы соединений *R*, описывается выражением:

$$r_{ij} = \sum_{s=1}^{M} q_{is} q_{js} w_s f_s, \qquad (1.4)$$

где $w_s$ – коэффициент $(0<w_s<=1)$, отражающий особенности $s$-й цепи; $f_s$ – коэффициент учета размера цепи. Задание соединений цепи полным графом, как это было сделано выше, соответствует выбору $w_s = f_s = 1$.

Перед размещением элементов каждая цепь заменяется полным графом, однако ребро графа получает вес $2/p_s$, где $p_s$ – размер цепи. Данная оценка определяет вероятность построения элементарного соединения между парой выводов цепи при условии равновероятного выбора любого из возможных деревьев.

Иногда оценка связности элементов по отдельной цепи выполняется с использованием величины fs=l/(ps-1). Эта оценка принимает значение 1 при

ps=2. С ростом размера цепи «степень связи» элементов, принадлежащих одной цепи, монотонно уменьшается.

В алгоритме размещения Куртзберга $f_s = (p_s + \lambda)/p_s$, что позволяет различать вклад цепей различного размера в весовую оценку $r_{ij}$.

Коэффициент $W_s$ может быть сделан относительно большим у цепей, для которых существенно время передачи сигнала.

При использовании в программах решения задач конструирования матрицы соединений *R* можно воспользоваться ее симметричностью.

Под связностью схемы при ее описании ВГС (матрицей *R)* понимается величина

$$S = \frac{1}{2}\sum_{i=1}^{n}\sum_{j}^{n} r_{ij} = \sum_{i=1}^{n}\sum_{j>1}^{n} r_{ij} \qquad (1.5)$$

При описании схемы ГЭК (матрицей *Q)* под связностью будем понимать суммарное число элементарных соединений:

$$S = \sum_{j=1}^{M}(\rho'_j - 1) = \sum_{j=1}^{M}\rho'_j - M\,, \qquad (1.6)$$

где *M* – число цепей.

Очевидно, что когда все цепи имеют размер *2*, значения связности схемы при ее описании ВГС и при описании схемы ГЭК совпадают.

В дальнейшем при рассмотрении алгоритмических методов решения основных задач конструирования используются различные модели для описания схем. Как правило, большинство из рассматриваемых алгоритмов могут при соответствующей модификации применяться для разных способов описания схем. Вместе с тем излишнее усложнение описания может привести к слишком сложным вычислительным процедурам и к большим затратам машинного времени, не оправданным с точки зрения конечного результата. В связи с этим сделаем несколько замечаний.

Описание коммутационной схемы с помощью ГКС является наиболее полным и точным и в одной из эквивалентных ему форм входит составной частью в исходную информацию для систем автоматизированного конструирования. Данная модель непосредственно используется при алгоритмическом решении задач трассировки соединений на ЭВМ. На основании этой же модели, как было показано выше, могут быть получены другие, более простые модели описания схемы. Модель схемы в виде ГКС должна иметь преимущества при решении ряда задач размещения, в которых игнорирование размеров элементов и положения их выводов является необоснованным. Сюда, прежде всего, относятся задачи размещения разногабаритных элементов, задачи размещения, в которых среднее расстояние между элементами на коммутационном поле сравнимо с размерами самих элементов, задачи, в которых внешние выводы существенно распределены по периферии схемы, и др. В этих случаях отождествление элементов с геометрическими точками (ГЭК или ВГС) приводит к значительным погрешностям уже на этапе постановки задачи оптимизации по тому или иному критерию (суммарной длине соединений, числу пересечений и т. п.). В связи с этим, использование более сложных и точных алгоритмов в таких ситуациях бессмысленно. Для решения задачи размещения целесообразно использовать упрощенные модели описания схем (ГЭК или ВГС) для выбора начального размещения и более точные (ГЭК и ГКС) при получении окончательного размещения.

Для задач компоновки элементов в узлы, учитывая специфику используемых при ее решении критериев и ограничений (числа межузловых связей, числа узлов и т. д.), информация о точном расположении выводов на элементах и узлах практически несущественна. Поэтому модели описания схем в виде ГЭК или ВГС в данном случае наиболее естественны, причем с точки зрения адекватности модели физическому содержанию задачи преимущество имеет представление схемы с помощью ГЭК.

С точки зрения реализации вычислительных процедур, наиболее просто описание схемы с помощью ВГС (матрицы соединений *R)*.

## 1.2 Алгоритмы размещения

### 1.2.1 Задача размещения как задача квадратичного назначения

Пусть даны элементы $e_1, e_2, \ldots, e_n$ и для каждой пары элементов заданы весовые коэффициенты $r_{ij}(i, j=1,2, \ldots, n)$, определяющие «степень связи» элементов друг с другом. Таким образом, считаем, что схема задана матрицей соединений $R=||r_{ij}||_{n\times n}$.

Пусть также имеется некоторый фиксированный набор позиций для размещения элементов $l_1, l_2, \ldots, l_m (m \geq n)$. Будем полагать, что $m=n$. Если $m>n$, то можно ввести $m-n$ фиктивных элементов, не связанных с остальными. Определим расстояние $d_{ij}$ между парами позиций. В любом случае, если на коммутационном поле фиксированы позиции для размещения элементов, то можно задать матрицу соединений $D=||d_{ij}||_{n\times n}$ , в которой элемент $d_{ij}$ равен расстоянию между центрами позиций $l_i$ и $l_j$. Очевидно, что матрица $D$– симметрическая с нулевой диагональю $(d_{ii}=0, i=1,2, \ldots, n)$.

Произвольное размещение элементов в позициях представляет собой некоторую перестановку $p=p(1), \ldots, p(i), \ldots, p(n)$, где $p(i)$ задает номер позиции, присвоенный $i$-му элементу. Таким образом, всего имеется $n!$ различных вариантов размещения элементов.

Рассмотрим задачу минимизации суммарной длины соединений (СДС) при следующих предположениях. Соединения будем считать условно исходящими из геометрических центров элементов. Кроме того, предполагается совпадение центров элементов и позиций. Как правило, при решении задачи размещения необходимо учитывать предварительное закрепление некоторых элементов в позициях и соединения элементов с внешними выводами. Сопоставляя внешним выводам элемент $e_0$ и фиксируя расположение элементов, получим упрощенное представление

коммутационного поля. Очевидно, что длина соединений между элементами $e_i$ и $e_j$ оценивается величиной $r_{ij}d_{p(i)p(j)}$. Обозначив через $L_s$ множество всех фиксированных элементов, включая элемент $e_0$; тогда суммарная взвешенная длина соединений элемента $e_i$ с элементами из $L_s$ оценивается по формуле

$$a_{ip(i)} = \sum_{s \in L_s} r_{ij} d_{p(i)s} \quad , \tag{1.7}$$

где $d_{p(i)s}$–расстояние между элементом $e_i$, находящимся в позиции $p(i)$, и элементом $e_s$.

Учитывая вышесказанное, а также симметричность матриц $R$ и $D$, выражение для суммарной взвешенной длины соединений при произвольном размещении запишется:

$$F(p) = \frac{1}{2}\sum_{}^{n}\sum_{}^{n} r_{ij} d_{p(i)p(j)} + \sum_{}^{n} a_{ip(i)} \quad , \tag{1.8}$$

Таким образом, задача размещения по критерию минимизации СДС состоит в минимизации функционала (1.8) на множестве перестановок $p$. Данная задача является вариантом общей математической модели, впервые рассмотренной Купмансом и Бекманом и получивший название задачи квадратичного назначения [ ].

В матрице переменных $X=||x_{ij}||_{n\times n}$, строчки соответствуют элементам, а столбцы – позициям. Переменная $x_{ij}=1$, если элемент $e_i$ находится в $l_i$ позиции, и $x_{ij}=0$ в противном случае. Легко установить взаимно-одназначное между матрицей $X$ и некоторой перестановкой элементов в позициях. Поэтому $X$ называют перестановочной матрицей или матрицей назначений. Очевидно, что матрица $X$ содержит в каждом столбце $j$ и в каждом строке $i$ одну единицу:

$$\sum_{i=1}^{n} x_{ij} = 1, i = 1,2,...,n$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ j = 1,2,\ldots, n\ .$$

С учётом введённых обозначений задачу размещения можно записать в форме обобщённой задачи квадратичного назначения: найти матрицу $X_0$, для которой

$$F(X_0) = \min\left(\frac{1}{2}\sum_{i,\,j=1}^{n}\sum_{s=1}^{n} r_{ij} d_{ks} x_{ik} x_{js} + \sum_{i,k=1}^{n} a_{ik} x_{ik}\right) \qquad (1.9)$$

Следует, отметить, что линейный член в (1.9) представляет собой функционал обычной (линейной) задачи о назначении. Последняя формулируется следующим образом. Пусть задана некоторая матрица $A=||a_{ij}||_{n\times n}$, строчки которой соответствуют некоторым объектам, а столбцы местам их назначения. Элемент $a_{ij}$ оценивает некоторую условную стоимость назначения объекта $i$ на место $j$. Задача состоит в нахождении такого назначения объектов по местам (без повторений), при котором суммарная стоимость всех назначений минимальна. В другой терминологии задача сводится к минимизации

$$\sum_{i=1}^{n} a_{ip(i)}$$

на множестве перестановок $P=\{p\}$, $A=||a_{ij}||_{n\times n}$ – заданная матрица коэффициентов.

К задаче линейного назначения сводится ряд частных задач оптимизации, возникающих при размещении элементов и при распределении их выводов. Задача размещения несвязанного множества элементов эквивалентна задаче линейного назначения.

Для задачи линейного назначения существуют эффективные точные методы решения. Одним из наиболее распространенных является венгерский алгоритм, позволяющий получить решение для n>100. Задача квадратичного назначения – существенно сложнее, поскольку «стоимость» назначения каждого элемента зависит от назначений всех связанных с ним элементов.

Для решения задачи квадратичного назначения предложен ряд алгоритмов, основанных на методе ветвей и границ.

Основная идея метода ветвей и границ состоит в разбиении всего множества допустимых решений задачи на некоторые подмножества, внутри осуществляется упорядоченный просмотр решений с целью выбора оптимального. Для всех решений, входящие в выделенные подмножества, вычисляется нижняя граница минимального значения целевой функции. Как только нижняя граница становится больше значения целевой функции для наилучшего из ранее известных решений, подмножество решений соответствующее этой границе, исключается из исходной области решений. Это обеспечивает сокращение перебора. Процесс поиска, сопровождаемый разбиением поля решений и вычислением нижних границ продолжается до тех пор, пока не будут исключены все решения, кроме оптимального.

Различия модификации общего метода применительно к задаче размещения (квадратичного назначения) отличаются способами расчёта нижних границ функционала (4.8) и способами разбиения поля решений.

Пусть имеется множество элементов $e_1, \ldots, e_i, \ldots, e_n$ и множество позиций $l_1, \ldots, l_i, \ldots, l_n$. Для описания процесса поиска решений введём дерево решений. Ребра первого яруса дерева пусть соответствуют назначениям элемента $e_1$, рёбра второго яруса –возможным вариантам назначения элемента $e_2$ в свободные позиции и т.д.

На рисунке 1.4 показана часть дерева полного дерева решений, в котором отмечены возможные назначения при условии, что элемент $e_1$ назначен в позицию $l_j$.

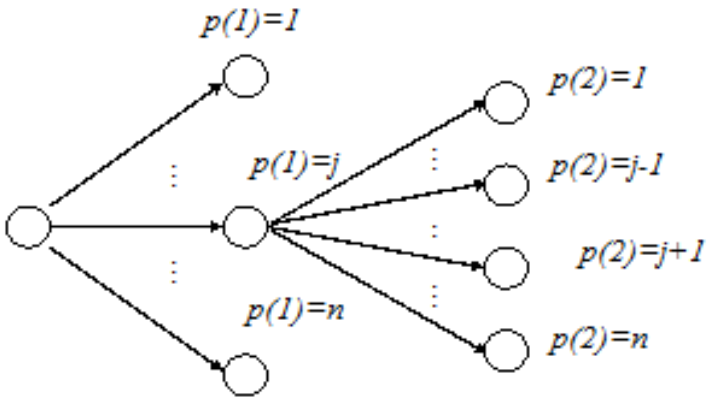


Рисунок 1.3 - Часть дерева решений

Произвольному размещению элементов соответствует в полном дереве некоторый путь, исходящий из начальной вершины. Для каждой вершины дерева можно рассчитать нижнюю границу целевой функции для множества путей (размещений), связанных с этой вершиной. Если эта граница больше значения целевой функции для известного размещения, то дальнейшее продвижение по дереву в данной вершине прекращается, поскольку оно приводит к заведомо не оптимальному решениям.

Частичным размещением $q$, будем называть назначение элементов множества $E_k=\{e_1, \ldots, e_i, \ldots, e_k\}$ в позиции множества $L_k=\{l_1, \ldots, l_j, \ldots, l_k\}$ в соответствии с правилом $j=q(i)$, $k \le n$. Ясно, что $q \subset p$ , где $p$–некоторое размещение $n$ элементов, в котором $k$ имеют размещение $q$. Частичному размещению $q$ соответствует в дереве решений путь, проходящий через вершины отдельных назначений.

Если в процессе продвижения по дереву нижние границы не рассчитываются и соответственно не производится усечений дерева, то метод ветвей и границ сводится к полному перебору.

Можно выделить следующие способы а отсечения ветвей [].

1) Сравнение оценки (нижней – $F_H$ или верхней – $F_B$ границы) со значением целевой функции для уже найденного (опорного) решения $F_{ОП}$. Действительно, если при решении задачи на минимум целевой функции $F_H > F_{ОП}$, то подмножество путей (размещений) не содержит оптимального варианта, и соответствующая вершина дерева решений

отсекается. Опорное решение можно получить приближенным алгоритмом заранее либо в ходе решения задачи методам ветвей и границ. При разбиении пространства решений по методу в глубину с возвращением опорное решение получается на более ранних этапах построения дерева решении, чем при методе в ширину.

2) Сравнение двух оценок. Такое отсечение выполняется, если, например, в задаче на минимум целевой функции, для подмножества вариантов соответствующей вершины можно найти оценку снизу $F_H$ и сверху $F_{ОП}$. Тогда, если для некоторого подмножества путей (размещений) , окажется что $F_H \geq F_B$ , то ветвление в соответствующей вершине прекращается.

Прекращение ветвления, если соответствующее подмножество не содержит оптимального решения, можно установить, во – первых, «В особых точках» по значениям оценочной функции, например нижней границы, и во – вторых, если известен оптимальный среди вариантов этого подмножества. Чем точнее будет получена оценка, т.е. чем ближе она будет к значению целевой функции для оптимального варианта подмножества, тем меньшее количество вершин дерева решений будет построено и исследовано.

Точность оценки зависит от принципа разбиения множества вариантов на подмножества и виды оценочной функции или способа ее вычисления. Следовательно, если возможно использовать несколько принципов разбиения, необходимо выбирать тот, при котором оценки более точны. Как правило, число отсечений тем больше, чем сильнее отличаются оценки подмножеств.

Таким образом, лучшими являются тот принцип разбиения и такая оценочная функция, при которых разность между оценками подмножеств наибольшая, а сами оценки вычисляются с наибольшей точностью в смысле их близости к значению целевой функции для оптимального варианта подмножества. Число отсечений зависит и от способа ветвления, хотя никаких точных оценок и рекомендаций узнать нельзя.

Рассмотрим *основные способы ветвления*.

1) Разбиение множества вариантов на полмножества по методу *« в ширину»* и выбор вершины ветвления по минимуму (максимуму) оценочной функции. Сначала выполняется разбиение всего множества вариантов, т.е. строятся на следующем от корня уровне все или часть вершин его потомков. Затем на каждом шаге выбирается вершина ветвления по минимуму нижней границы (максимуму верхней для задачи на максимум целевой функции) и разбивается соответствующее ей подмножество вариантов. В ходе ветвления используется, если это возможно, тот или иной способ отсечения ветвей и вершин. Процесс выбора вершин и ветвления повторяется для всех висячих вершин, т.е еще не вариантов решения задачи.

2) Разбиение множества вариантов по методу *«в глубину с возвращением*», т. е. последовательное построение ветвей. Строят полностью одну ветвь дерева решений, т.е. находят один вариант решения задачи. Полученное для этого варианта значение целевой функции может использоваться как отсекающая оценка. Далее ветвление выполняется последовательно от построенной ветви, начиная с вершины, предшествовавшей конечной. При этом, если не происходит отсечение, новая ветвь достраивается до конца. Процедура повторяется по отношению к вершинам новой ветви. После построения всех возможных ветвей, проходящих через вершину, сопоставленную множеству путей (размещений ), отыскивается следующая вершина и процесс повторяется. При последовательном способе ветвления построение дерева решений может выполняться, начиная с левой ветви – слева направо, или с правой – справа налево. Количество построенных вершил дерева решений зависит от очередности развития ветвей.

3) *Комбинация двух рассмотренных способов*. Простейшая комбинация заключается в следующем. Строится одна ветвь. Далее развитие дерева решений происходит по методу «в ширину», полученное опорное решение используется для отсечения, выбор вершины ветвления выполняется по

минимуму нижней границы в задачах на минимум целевой функции (по максимуму верхней).

В ходе решения, если возможно, уточняется значение отсекающей оценки. Оптимальное решение будет найдено, когда в задаче на минимум целевой функции значение для некоторой конечной вершины меньше нижней границы $F_H$ для всех висячих вершин и меньше значения целевой функции F для всех остальных конечных вершин (в задаче на поиск максимума целевой функции соответственно «больше»).

Рассмотрим способы вычисления нижних границ функционала (1.8) при частичных размещениях $q$. Обозначим первый член в выражении $F(p)$ через $w(p)$, а второй через $v(p)$:

$$F(p) = w(p) + v(p) \tag{1.10}$$

Нижняя граница для $F(p)$ получена суммированием нижних границ для $w(p)$ и $v(p)$. Обозначим их соответственно $b_F$, $b_w$ и $b_v$.

Для вычисления $b_w$ воспользуемся следующим свойством: если $r=(r_1,r_2,\ldots, r_m)$ и $d=(d_1, \ldots, d_2, \ldots, d_m)$ – два вектора, то минимум скалярного произведения $r$ и $d$, т.е. минимум $\sum_{j=1}^{m} r_i d_{p(i)}$ $\sum_{i=1}^{m} r_i d_{p(i)}$ на множестве всех перестановок p, соответствует расположению составляющих вектора $r$ в возрастающем порядке, а составляющих вектора $d$ в убывающем.

Нижняя граница для $F(p)$

$$b_F = b_w + b_v \tag{1.11}$$

должна быть соотнесена начальной вершине дерева решений.

Пусть теперь имеется некоторое частичное размещение $q$, определяющее назначение элементов из множества $E_k$ в позиции из множества $L_k$. Тогда можно

легко рассчитать вклад в общую длину соединений уже размещенных элементов и вычислить новую нижнюю границу соединений.

Запишем (1.8) для *F(p)* в следующем виде:

$$F(p) = F(q) + \sum_{i \in E_k} \sum_{s \in E'_k} r_{is} d_{q(i)p(s)} + \sum_{i \in E'_k} \sum_{s \in E'_k} r_{is} d_{q(i)p(s)} + \sum_{i \in E'_k} a_{ip} \quad (1.12)$$

где $E'_k$–множество не размещенных элементов для частичного размещения $q \subset p$.

Структуру (1.11) можно записать в компактной форме:

$$F(p) = F(q) + w_q(p) + v_q(p) + u_q(p), \quad (1.13)$$

где *F(q)*- длина межсоединений размещённых элементов; члены $w_q(p)$ и $v_q(p)$ аналогичны членам выражения (1.10) и соответствуют межсоединениям неразмещенных элементов; $u_q(p)$ представляет собой суммарную длину соединений размещенных элементов с неразмещенными.

Поскольку *F(q)* полностью определяется частными размещениями *q*, для вычисления новой нижней границы для *F(p)* необходимо рассчитать нижние границы для остальных членов выражения (1.13) . Границы для $w_q(p)$ и $u_q(p)$ рассчитываются по матрицам R и *D*, а нижняя граница для $v_q(p)$ определяется решением задачи линейного назначения неразмещенных элементов $E'_k$ в незанятые позиции $L'_k$.

Выбор направления поиска в дереве решений может быть организован различными способами. Как правило, очередное ветвление производится из вершины дерева с наименьшим значением нижней границы.

Далее поиск продолжается в соответствии с общей схемой метода ветвей и границ. Процесс заканчивается тогда, когда в просмотренной части дерева решений отсутствуют вершины, для которых нижняя граница меньше, чем у наилучшего из известных решении. Это решение может быть либо получено

одним из приближенных алгоритмов, и в этом случае доказывается оптимальность этого решения, либо определено в процессе поиска путем завершения некоторого частичного размещения *q*. В последнем случае в дереве решений определяется путь, соответствующий оптимальному размещению.

Расчет нижних границ при частичных размещениях осуществляется с целью усечения дерева решений. Естественно, что чем точнее эти границы, тем существеннее усечение. В связи с этим укажем более точный способ вычисления нижней границы для *F(p)*.

Пусть $r_i$ – вектор-строка матрицы $R$ без элемента $r_{ii}=0$, а $d_j$,– вектор-строка матрицы $D$ без элемента $d_{ii}=0$. При назначении элемента $e_i$ в позицию $l_j$ минимальная длина его связей с остальными элементами равна минимуму скалярного произведения векторов $r_i$_и $d_j$. Обозначим его через $r_i \cdot d_j$. Образуем матрицу $A'=||a'_{ij}||_{n\times n}$, в которой элемент $a'_{ij}$ представляет оценку назначения элемента $e_i$ в позицию $l_j$:

$$a'_{ij} = r_i \cdot d_j + a_{ij}, \qquad (1.14)$$

где $a_{ij}$ – элемент линейной части *v(p)* функционала (1.8) или (1.9).

Теперь нижняя граница для *F (p)* может быть определена решением задачи линейного назначения для матрицы *A*.

Распространение этого способа на вычисление границ при частичном размещении *q* производится путем образования матрицы $A'_k = ||a'_{ij}||_{(n-k)\times(n-k)}$ для неразмещенных элементов из множества $E'_k$.

### 1.2.2 Непрерывные модели конструкций

Рассмотрим выражение для суммарной взвешенной длины:

$$F = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij} d_{ij}\ , \qquad (1.15)$$

где $d_{ij}$ – расстояние между точками $(x_i, y_i)$ и $(x_j, y_j)$, определяющими положение элементов $e_i$ и $e_j$.

Функцию $F$ можно считать функцией $2n$ переменных: $x_i$ $(i=1, 2, \ldots, n)$ и $y_i$ $(i=l, 2, \ldots, n)$. Если $d_{ij}=d(x_i, y_i, x_j, y_j)$ рассчитывать по

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

$$d_{ij} = | x_i - x_j | + | y_i - y_j |$$

функция (1.8) является кусочно-выпуклой и потому для ее минимизации могут быть в принципе использованы аналитические методы. Для отыскания минимума приравняем частные производные по нефиксированным переменным нулю и после очевидных упрощений получим систему алгебраических уравнений:

$$\begin{aligned} \sum_{j=1}^{n} \frac{\partial d(x_i, y_i, x_j, y_j)}{\partial x_i} = 0 \quad i = 1,2, ..., n \\ \sum_{j=1}^{n} \frac{\partial d(x_i, y_i, x_j, y_j)}{\partial y_i} = 0 \quad i = 1,2, ..., n \end{aligned} \qquad (1.16)$$

Очевидно, что если все переменные независимы, то (1.16) имеет тривиальное решение: $x_i=const$ $(i=1, 2, \ldots, n)$, $y_i=const$ $(i=l, 2, \ldots, n)$. Следовательно, данная модель не может быть непосредственно использована для размещения элементов, поскольку все элементы «стягиваются» в точку. Возможно несколько путей корректировки этой модели.

Один из них состоит в предварительной фиксации положений некоторых элементов. Такими элементами могут быть внешние выводы узла или

элементы, положение которых задано конструктором. Предположим, что не фиксированы положения первых *l* элементов, а положения *n-l* элементов фиксированы.

Рассмотрим более подробно случай задания расстояния по формуле

$$d_{uj} = (x_i - x_j)^2 + (y_i - y_j)^2$$

Тогда (1.16) сводится к системе линейных алгебраических уравнений:

$$\sum_{j=1}^{n} r_{ij}(x_i - x_j) = 0 \quad i = 1,2,...,l$$

$$\sum_{j=1}^{n} r_{ij}(y_i - y_j) = 0 \quad i = 1,2,...,l$$

Запишем ее в следующем виде:

$$\begin{aligned} \sum_{j=1}^{n} r_{ij} x_i - \sum_{j=1}^{l} r_{ij} x_j &= \sum_{j=l+1}^{n} r_{ij} x_{ij} \quad i = 1,2,...,l \\ \sum_{j=1}^{n} r_{ij} y_i - \sum_{j=1}^{l} r_{ij} y_j &= \sum_{j=l+1}^{n} r_{ij} y_{ij} \quad i = 1,2,...,l \end{aligned} \qquad (1.17)$$

Определим матрицу коэффициентов A = ||aij||l×l, в которой $a_{ij} = \sum_{j=1}^{n} r_{ij}$ , aij =---rij . Отметим, что матрица A – симметрическая, ввиду симметричности матрицы соединений R. Введем векторы b=(b1, …, bi, …, bl) и c=(c1 ..., ci,…,cl), составляющие которых равны:

$$b_i = \sum_{j=l+1}^{n} r_{ij} x_j,\ i = 1,2,...,l$$
$$c_i = \sum_{j=l+1}^{n} r_{ij} y_j,\ i = 1,2,...,l \qquad (1.18)$$

Тогда (1.17) может быть представлена в стандартной матричной форме:

$$AX = b$$
$$AY = c \qquad (1.19)$$

где $X = (x_1,\ldots,x_i,\ldots,x_l)$ и $Y = (y_1,...,y_i,...,y_l)$ – векторы неизвестных.

Для решения (1.19) могут быть использованы известные вычислительные методы. В. М. Помазановым исследованы условия сходимости метода Гаусса – Зейделя для данной системы и даны рекомендации по предварительной фиксации некоторых элементов с целью получения достаточной равномерности расположения элементов [ ].

Рассмотренную точечную модель размещения элементов можно расширить, задав геометрию отдельных элементов. В частности, при размещении кристаллов интегральных схем на подложке учитывается расположение выводов элементов и внешних выводов микросхем.

Аналогичные модели могут быть построены и для других способов расчета расстояний $d_{ij}$. Однако в этом случае соответствующие системы уравнения (1.17) оказываются значительно более сложными.

Основными недостатками рассмотренного метода являются необходимость предварительной фиксации отдельных элементов, возможность перекрытия конфигураций элементов в получаемом размещении и недостаточная равномерность их расположения. Поэтому данный метод рекомендуется применять для определения грубого начального варианта размещения, который впоследствии должен корректироваться.

Значительно большая степень равномерности может быть достигнута при введении в модель размещения ограничений на расположение элементов. В теоретическом плане представляет интерес градиентный метод Линского в

котором задача размещения сводится к задаче нелинейного программирования, и работа Холла, в которой решается задача на условный экстремум функции (1.8) при квадратичных ограничениях на расположение элементов.

После очевидных преобразований (1.17) получим следующие выражения:

$$x_i = \frac{\sum_{j=1}^{n} r_{ij} x_j}{\sum_{j=1}^{n} r_{ij}} \quad i = 1,2,...,n$$

$$y_i = \frac{\sum_{j=1}^{n} r_{ij} y_j}{\sum_{j=1}^{n} r_{ij}} \quad i = 1,2,...,n \qquad (1.20)$$

Выражения (1.20) аналогичны известным формулам вычисления центра масс системы материальных точек. В ряде алгоритмов размещения используется последовательный процесс установки элемента $e_i$ *в* «центр масс» связанных с ним элементов $e_i$, которые в этот момент считаются неподвижными. В качестве массы элемента $e_j$ принимается величина $r_{ij}$ – число соединений $e_j$ с элементом $e_i$. Масса элемента $e_i$ оценивается суммарным числом его соединений с остальными элементами.

Использование (1.20) для определения положения элементов представляет меньшие вычислительные трудности по сравнению с решением (4.19) и потому имеет явные преимущества при решении задач размещения с большим числом элементов $n$. Кроме того, при этом легче учитываются ограничения на расположение элементов и их размеры. Аналогичные модели могут быть использованы и для других способов расчета расстояний между элементами.

Отметим, что использование данного метода предполагает задание начального варианта размещения и введение средств предотвращения группировки элементов в отдельных областях коммутационного поля. Развитием описанной выше механической аналогии можно считать метод силовых функций. Он основан на сведении задачи размещения к задаче отыскания состояния равновесия системы материальных точек, на которые действуют силы притяжения и отталкивания. Непрерывные модели и механические аналогии представляют определенный интерес для решения задач размещения элементов, в которых набор позиций заранее не задан. Однако их непосредственное использование в практических задачах размещения малоэффективно, поскольку они, как правило, не обеспечивают равномерности расположения элементов на коммутационном поле.

### 1.2.3 Алгоритмы конструкторского проектирования

Среди алгоритмов конструкторского проектирования (КП) выделяют два основных класса: конструктивные и итерационные.

*Конструктивные алгоритмы* формируют проектное решение за ряд последовательных шагов:

— выбирается один элемент схемы рассматриваемого уровня;

— к выбранному элементу по определенным правилам присоединяется второй;

— к полученному комплексу элементов добавляется третий и т.д.

Алгоритмы, использующие подобную методологию, называются последовательными. Алгоритмы, в которых формируются несколько групп элементов в пределах одного шага, называются параллельными.

*Итерационные алгоритмы* требуют задания начального приближения решения задачи КП, которое затем улучшается. Начальное решение задается инженером-проектировщиком (пользователем САПР) или является результатом работы конструктивного алгоритма.

Для решения задач конструирования ЭУ с использованием алгоритмов КП необходимо сформулировать задачи компоновки, размещения и трассировки и решить их как задачи поиска экстремума целевой функции.

Для этого нужно, во-первых, формализовать понятие «оптимальная» компоновка, «оптимальное» размещение, «оптимальная» трассировка; во-вторых, решить задачу, уже имеющую математическую формулировку. Решение задачи конструкторского проектирования строгими математическими методами может быть выполнено после того, как задача поставлена. Сама постановка задачи ведется с учетом заданной коммутационной схемы и цели компоновки и размещения элементов на коммутационном поле, трассировки монтажных соединений.

Процедура постановки задачи оптимизации носит неформальный характер и включает этапы: выбор целевой функции и управляемых параметров; назначение ограничений; нормирование параметров.

Сложность постановки оптимизационных проектных задач обусловлена наличием у проектируемых объектов нескольких выходных параметров, которые могут быть критериями оптимальности, но в задаче целевая функция должна быть одна. Другими словами, проектные задачи являются многокритериальными, и возникает проблема сведения многокритериальной задачи к однокритериальной, что называется сверткой векторного критерия. Среди выходных параметров всегда можно найти пары таких, что улучшение одного из них приводит к ухудшению другого. Такие параметры называют конфликтными, т.е. при оптимизации ТО невозможно улучшение всех выходных параметров одновременно.

В зависимости от того, каким образом выбираются и объединяются выходные параметры в скалярной функции качества различают *критерии оптимальности*:

1) *Частный критерий* применяется, когда среди выходных параметров можно выделить один основной, наиболее полно отражающий эффективность проектируемого объекта. Этот параметр принимают за целевую функцию.

Например, грузоподъемность для транспортных средств, мощность для энергетического объекта. Условия работоспособности остальных выходных параметров относят к ограничениям задачи. Достоинством является простота, недостатком такого подхода – большой запас работоспособности можно получить только по основному параметру, который принят в качестве целевой функции, а другие выходные параметры не будут иметь запасов.

И комплексные критерии, которые в свою очередь делятся на:

2) *Аддитивный критерий* объединяет (свертывает) все выходные параметры (частные критерии) в одну целевую функцию, представляющую собой взвешенную сумму частных критериев

$$F(X) = \sum_{j=1}^{m} \omega_j y_j(X), \qquad (1.21)$$

где $\omega_j$– весовой коэффициент, $m$ – число выходных параметров. Функция подлежит минимизации, при этом если условие работоспособности имеет вид $yj > Tj$ , то ωj < 0. Недостатки аддитивного критерия – субъективный подход к выбору весовых коэффициентов, которые выбираются либо самим инженером-проектировщиком, либо группой экспертов, и в процессе оптимизации остаются неизменными и не учет требований технического задания на проектирование. Действительно, в (1.21) не входят нормы выходных параметров.

3) *Мультипликативный критерий* используется, когда отсутствуют условия работоспособности типа равенств и выходные параметры не могут принимать нулевые значения. Целевая функция имеет вид

$$F(X) = \prod_{j=1}^{m} (y_j(X))^{\alpha_j} \qquad (1.22)$$

Нетрудно видеть, что если прологарифмировать (1.22), то мультипликативный критерий превращается в аддитивный. Достоинством является отсутствие нормирования выходных параметров. Недостатки те же, что у аддитивного критерия. Общим недостатком двух вышеприведенных

критериев является улучшение одних выходных параметров за счет недопустимого ухудшения других.

4) Максиминный (минимаксный) критерий - наиболее предпочтительный критерий при проектировании ТО. Он позволяет достичь наилучшего удовлетворения условий работоспособности. В качестве целевой функции принимается выходной параметр, наиболее неблагополучный с позиций выполнения условий работоспособности. Для оценки степени выполнения условия работоспособности j-го выходного параметра вводят запас работоспособности этого параметра Sj и этот запас можно рассматривать как нормированный j-й выходной параметр. Например (здесь и далее для лаконичности изложения предполагается, что все выходные параметры приведены к виду, при котором условия работоспособности становятся неравенствами в форме yj < TTj ):

$$S_j = ( TT_j - y_j ) / TT_j$$

или

$$S_j = a_j(( TT_j - y_{jном}) / \delta_j - 1)) ,$$

где $y_{jном}$ – номинальное значение, $a_j$ – весовой коэффициент, $\delta_j$ – некоторая характеристика рассеяния $j$-го выходного параметра, например, трехсигмовый допуск. Тогда целевая функция в максиминном критерии есть

$$F(X) = \min_{j \in [1:m]} S_j (X).$$

Здесь запись [*1: m]* означает множество целых чисел в диапазоне от 1 до m.

Достоинством максиминного критерия является то, что на целевую функцию влияет только тот $y_j$ , который в данной точке $x$ является наихудшим с позиций выполнения требований технического задания.

Анализ процесса конструирования электронных устройств (ЭУ) на основе существующих методов можно проводить, используя разные точки зрения. На входе процесса проектирования ЭУ имеется функциональная схема, которая

содержит информацию о базовых элементах (в зависимости от уровня проектирования), о связях между элементами и внешних связях проектируемого ЭУ. Кроме того, задаются технологические параметры, например, при конструировании печатного узла – размер печатной платы, разрешенные для применения в данной разработке серии микросхем, шаг сетки трассировки и т.п. В результате проектирования должен получиться рисунок трассировки.

Общей целевой функцией Ф всего процесса проектирования печатной платы (ПП) следует считать число проведенных связей. Целевая функция F зависит от случайных входных параметров ζ (например, числа базовых элементов, числа задействованных выводов элементов схем и т.д.), так и от переменных, статистически устойчивых для класса разработок β (размеры ПП, серии микросхем, разрешенных для применения и т.п.). Хотя оптимальный размер ПП может быть определен с достаточной точностью как функция входных параметров, этот вопрос практического применения не нашел, так как изготовление и применения плат различных размеров в одной разработке связано с изменением технологической оснастки производства.

Действия над входными величинами (ζ, β) в процессе проектирования можно представить следующим образом. Обозначив совокупность используемых в САПР алгоритмов компоновки К, совокупность алгоритмов размещения Р и совокупность алгоритмов трассировки Т можно представить схему последовательной реализации основных алгоритмов, рисунок 1.5.

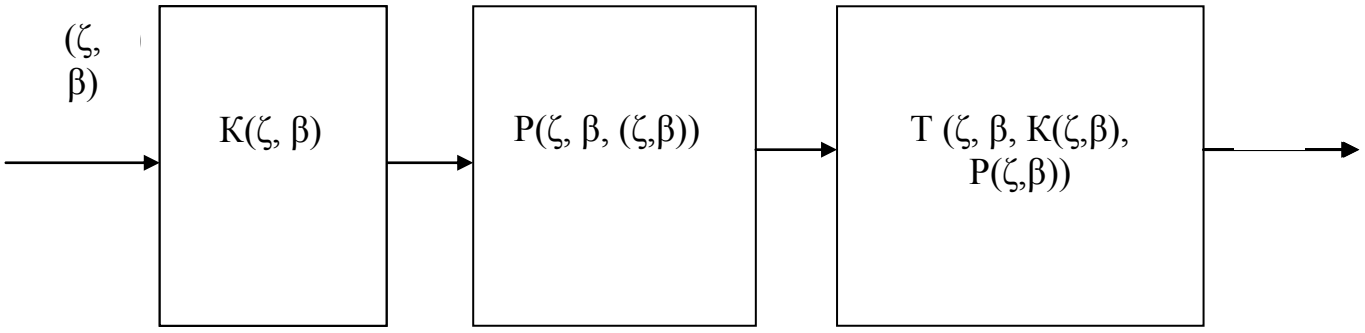



Рисунок 1.5 - Действия над входными величинами

Возможно использование алгоритмов совместного решения задач компоновки и размещения, размещения и трассировки, компоновки и трассировки (при этом трассировка выполняется на условном коммутационном поле).

Общая задача проектирования формулируется следующим образом: имея заданные условия β и случайные воздействия ζ, найти такую совокупность алгоритмов и критериев {К, Р, Т}которая обеспечивала бы получение максимального конечного значения целевой функции F. Таким образом, имеется задача о выборе и принятии решения в условиях неопределенности. Оптимизация решения на каждом шаге отдельно не всегда дает в сумме оптимальное решение, особенно, если на промежуточных этапах слабо учитывается конечный критерий. Указанный недостаток является причиной поиска связей между основными этапами КП.

Реализованные алгоритмы К, Р, Т в составе интегрированной САПР должны сымитировать действия конструктора, который решает одновременно все три задачи с оптимизацией. Это требует создания обширного математического обеспечения САПР, позволяющего варьировать различными алгоритмами. Полная алгоритмизация действий конструктора затруднительна, поэтому задача решается поэтапно, через выстраивание преемственных критериев всех этапов, поэтому при разработке САПР ставятся упрощенные задачи, т.е. на каждом шаге принятия решений приближаться к оптимальной траектории F. Очевидно, что само выделение трех этапов КП – компоновки, размещения и трассировки, направлено на снижение размерности общей задачи. Декомпозиция проводится следующим образом, что сначала осуществляется компоновка путем оптимальной группировки функциональных узлов, затем размещение компонентов выделенных узлов в их монтажном пространстве с учетом критериев, отражающих оптимальность последующего этапа – трассировки соединений.

### 1.2.4 Алгоритмы решения задачи размещения

Выше было дано определение задачи размещения на этапе конструкторского проектирования ЭУ, как задачи размещения конструктивов *i-го* уровня по позициям коммутационного поля конструктивов *i+1*-го уровня. Рассматривая элементы ЭУ в качестве конструкивов *i-го* уровня, а печатные платы в качестве конструктивов *i+1*-го уровня, решение задачи размещения элементов ЭУ должно определить такое их местоположения на коммутационном поле, при котором создаются наилучшие условия для решения последующей задачи трассировки соединений с учетом конструктивно-технологических требований и ограничений.

При решении задачи размещения элементов ЭУ находят применение следующие критерии размещения:

— суммарная длина всех связей;

— расстояние между элементами, соединенными наибольшим числом связей;

— число пересечений проводников на КП;

— длина наиболее длинных связей;

— число цепей с возможно более простой конфигурацией;

— число перегибов проводников;

— число межслойных переходов;

— параметры паразитных связей между элементами и проводниками;

— равномерность температуры по поверхности КП.

В большинстве случаев основным рассматриваемым критерием оптимальности при решении задачи размещения является минимизация суммарной длины соединений (СДС) (1.15), который косвенно учитывает другие из перечисленных критериев. Это обуславливается рядом факторов:

— уменьшение длин соединений улучшает электрические параметры схемы;

— как показывает практика, чем меньше суммарная длинна соединений, тем в среднем проще их реализация при трассировке;
— уменьшение суммарной длины соединений снижает трудоёмкость изготовления монтажных схем, особенно проводного монтажа.

Все известные и доведенные до практической реализации алгоритмы размещения являются приближенными, что оказывается достаточным, так как величина $d_{ij}$ в (1.15)– суть расстояния между позициями установки элементов (точками) а не истинные длины соединений, которые определяются при дальнейшей задачи трассировки. При размещении элементы представляются точками, являющимися геометрическим центром корпуса элемента, что значительно упрощает решение задачи, к тому же принципиальных алгоритмических трудностей при переходе от точек к элементам нет.

На рисунке 1.6 выделяют основные группы алгоритмов размещения:

1) Алгоритмы решения математических задач, являющиеся моделями задачи размещения.
2) Конструктивные алгоритмы начального размещения.
3) Итерационные алгоритмы улучшения начального варианта размещения.
4) Непрерывно-дискретные методы размещения.

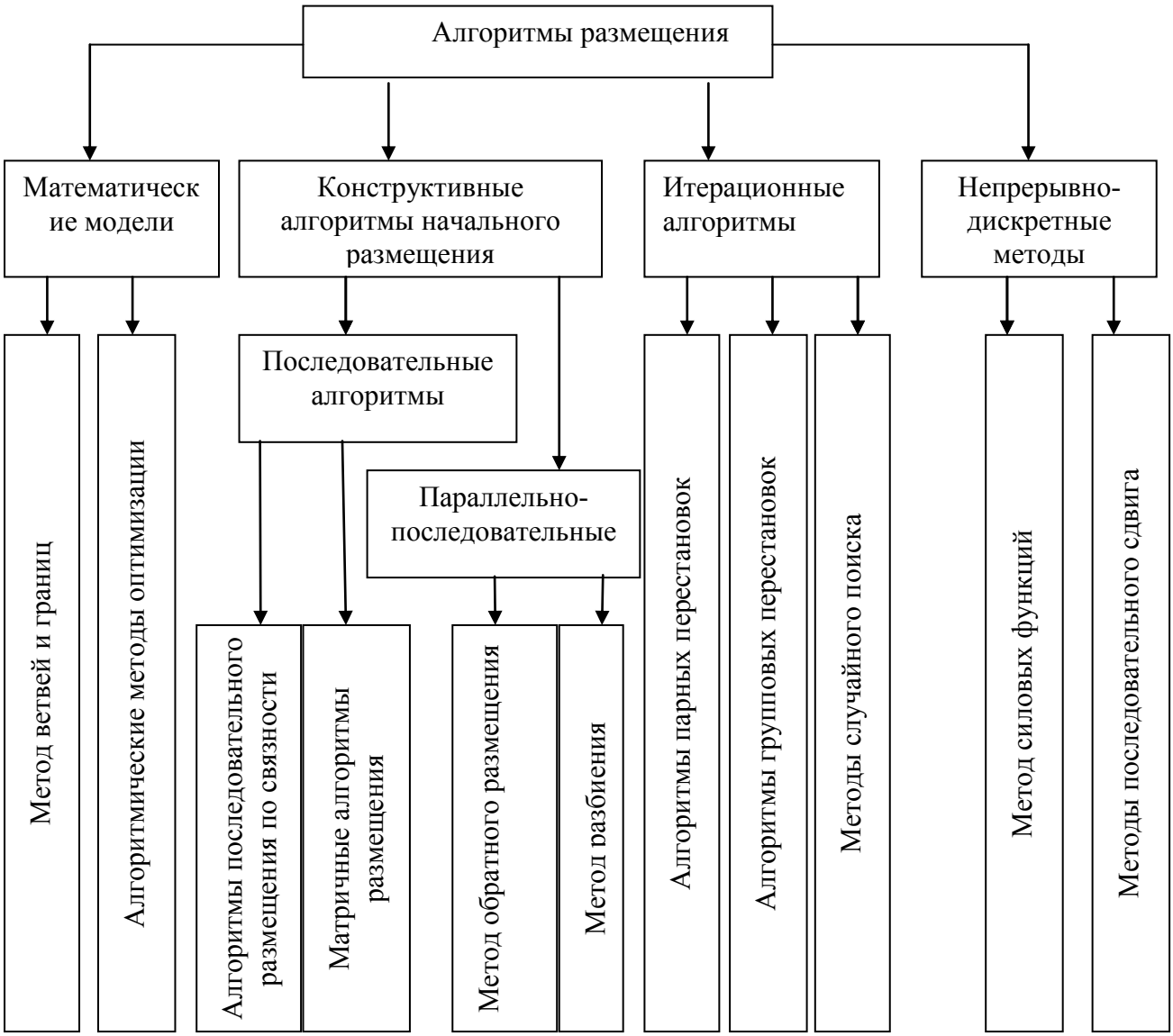



Рисунок 1.6 - Классификация алгоритмов размещения

К первой группе относятся, прежде всего, метод ветвей и границ для задачи квадратичного назначения, к которой при определённых упрощениях сводится задача размещения: набор позиций считается фиксированным, элементы рассматриваются как геометрические точки, схема соединений представляется взвешенным графом. Другой класс моделей связан с оптимизацией размещения, когда набор позиций для установки заранее не фиксирован.

Вторая и третья группы включают приближённые алгоритмы, в основном предназначенные для оптимизации размещения элементов в фиксированном наборе позиций.

Характерной чертой конструктивных алгоритмов является то, что они создают размещение. Тогда как итерационные алгоритмы предполагают задание начального размещения. Конструктивные алгоритмы используют последовательный или параллельно-последовательный процесс установки элементов в позиции при локальной оптимизации функции-критерия размещения.

В итерационных алгоритмах производится переразмещение элементов или их групп с целью минимизации выбранного критерия. Эти алгоритмы требуют существенных затрат машинного времени и используются для получения конечного размещения.

Основной областью применения непрерывно-дискретных методов размещения являются конструкции, в которых позиции для установки элементов заранее не фиксированы. Исходной базой для построения этих алгоритмов являются непрерывные модели.

Исходной информацией для размещения является схема соединений, параметры конструкции элементов и коммутационного поля.

**Алгоритмы последовательного размещения по связности.** Пусть $E=\{e_1, e_2, ..., e_n\}$ – множество элементов, подлежащих размещению, а $L=\{l_1\ l_2, ..., 1_n\}$ – множество позиций для их установки. Вводится $n$-шаговый процесс принятия решений, на каждом шаге которого выбирается один из неразмещенных элементов и помещается в одну 'из незанятых позиций. Структура любого последовательного алгоритма размещения определяется правилами выбора очередного элемента и позиции для его установки.

Пусть $E_k$ – элементы, размещенные до $k$-го шага, а $L_k$ – позиции, занятые этими элементами; $E'_к$ и $L'_k$ – соответственно неразмещенные элементы и незанятые позиции. Отметим, что перед началом размещения могут быть две ситуации: нет размещенных ранее элементов, внешние выводы узла (контакты, разъемы и т. п.) не закреплены (в этом случае в алгоритме должен быть особо определен способ установки первого элемента); имеется группа заранее размещенных элементов или закрепленных внешних выводов.

В основу большинства последовательных алгоритмов размещения положен эвристический принцип оптимизации целевой функции, сводящийся к выбору на данном шаге локально оптимальной позиции для одного из элементов при неизменности положения ранее размещенных элементов. Поскольку критерий минимума суммарной длины соединений в силу отмеченных ранее причин наиболее распространен, он и будет рассмотрен при описании алгоритмов данной группы. Вместе с тем, используемые в этих алгоритмах тактики размещения могут быть с некоторыми модификациями применены и для других критериев. В алгоритмах размещения по связности элемент и позиция выбираются независимо.

*Выбор элемента*. Любое правило выбора элемента для размещения основано на вычислении «меры связности» еще неразмещенных элементов с уже размещенными. Естественной мерой связности двух элементов $e_i$ и $e_j$ является количество соединений между ними, заданное в матрице соединений $R=|r_{ij}|_{n\times n}$.

Так, в алгоритме «попарных связей» Куртзберга для каждого неразмещенного элемента $e_i$ подсчитывается характеристика

$$c_i = \max_{e_j \in E_k} r_{ij} \qquad (1.23)$$

причем $r_{ij}$ засчитывается по формуле

$$r_{ij} = \sum_{s \in J_{ij}} \frac{p_s + \lambda}{p_s} w_s \qquad (1.24)$$

в которой $J_{ij}$ – множество цепей, связывающих элементы $e_i$ и $e_j$ $p_s$–размер цепи; $w_s$ – весовой коэффициент; λ. – целочисленный параметр.

Параметр λ позволяет дифференцировать вклад цепей различного размера. Чем больше значение λ, тем больше влияние цепей с малым значением $p_s$. Очевидно, что при λ=0 значение $r_{ij}$ (1.23) равно суммарному весу цепей между элементами $e_i$ и $e_j$. Поскольку всегда$P_2 \geq 2$, то λ ≥-1.

Очередным размещаемым элементом является элемент, имеющий максимальную характеристику (1.24), т. е. выбор элемента осуществляется по наибольшему числу связей с уже размещенными элементами.

Другое правило выбора основано на расчете для каждого неразмещенного элемента $e_i \in E'_k$ суммарной связности с уже размещенными элементы. Если задаётся матрица соединений *R*, то соответствующая характеристика

$$c_i = \sum_{e_j \in E_k} r_{ij} \qquad (1.25)$$

Если для схемы задан список цепей (комплексов), то аналогичная характеристика имеет вид

$$c_i = | J_i ( \bigcup_{e_j \in E_k} J_j ) | \qquad (1.25)$$

где $J_i$ – множество цепей связанных с элементом $e_i$. $C_i$ равна конъюнкции множества цепей элемента $e_i$ и множества цепей $E_k$. Выбор в данном случае осуществляется на основании максимального значения характеристики (1.24) и (1.25).

Можно использовать другое правило выбора элемента, основанное на оценке числа связей не размещённого элемента $ei \in E'_k$ как с размещенными, так и с неразмещенными элементами:

$$c_i = \sum_{e_j \in E_k} r_{ij} - \sum_{e_j \in E'_k} r_{ij} \qquad (1.26)$$

В этом случае выбирается элемент с максимальным значении $c_i$ (1.26). Заметим, что данный выбор аналогичен принципу максимальной конъюнкции - минимальной дизъюнкции, применяемому в алгоритмах компоновки.

К этому же типу относится характеристика относительной связности:

$$c_i = \sum_{e_j \in E_k} r_{ij} \Big/ \sum_{e_j \in E_k} r_{ij}$$

На очередном шаге алгоритма размещается элемент, имеющий максимальный коэффициент относительной связности.

В связи с тем, что при размещении элементов число соединений между отдельными группами элементов не может быть строго определено (оно зависит от способа последующей трассировки соединений), в некоторых алгоритмах используются вероятностные оценки для подсчета меры связности.

Выбранный для размещения элемент $e_{i_0}$. должен быть установлен в одну из незанятых позиций из множества $L'_k$- Эта позиция выбирается с учетом минимизации критерия размещения, в частности, МСД соединений. Естественно, что при последовательном процессе размещения может быть оценена лишь суммарная длина частичных монтажных соединений данного элемента $e_{i_0}$. с уже размещенными элементами $E_k$.

В принципе возможен подход, когда при установке элемента в позицию рассчитываются трассы соответствующих соединений. Длина этих соединений является критерием для выбора позиции. Однако большие затраты машинного времени делают этот подход нереальным, по крайней мере, при конструировании узлов с печатными соединениями, и ограниченно применимым при конструировании монтажных схем проводных соединений.

Учитывая вышесказанное, используют приближенные оценки длин монтажных соединений. Наиболее простая из них и наименее точная сводится к расчету расстояний между позициями.

**Метод обратного размещения.** В методе обратного размещения осуществляется предварительная оценка каждого из размещаемых элементов $e_1, e_2, \ldots, e_n$ и каждой свободной позиции $l_1, l_2, \ldots, l_n$, после чего все элементы размещаются одновременно.

Пусть дана матрица соединений $R=|r_{ij}|_{n\times n}$ и матрица расстояний между позициями $D=|d_{ij}|_{n\times n}$.

Этапы метода обратного размещения:

1) Для каждого элемента ei по матрице R найдем суммарное число соединений этого элемента с остальными элементами:

$$r_i = \sum_{j=1}^{n} r_{ij} \quad i = 1,2,\ldots,n \qquad (1.27)$$

2) Для каждой позиции $l_i$ по матрице $D$ найдем характеристику

$$d_i = \sum_{j=1}^{n} d_{ij} \quad i = 1,2,\ldots,n, \qquad (1.28)$$

определяющую суммарное расстояние этой позиции до остальных позиций.

Позиции в центральной части КП имеют меньшую характеристику $d_i$, чем позиции на периферии. Естественно, что центральные позиции наиболее благоприятны для размещения сильно связных элементов, т.е. элементов имеющих большее значение $r_i$ (1.27). Рассматривая с этой точки зрения выражение (1.15) и учитывая условия минимальности скалярного произведения $r \cdot d$ получаем следующее:

3) Упорядочить элементы по возрастанию характеристики

$r_i$ $(r_{i_1} \le r_{i_2} \le \ldots \le r_{i_n})$

4) Упорядочить позиции по убыванию характеристики $d_{ij}$ ($d_{i_1} \geq d_{i_2} \geq \ldots \geq d_{i_n}$)

5) Определить размещение $p(i_k)=j_k$, $k=1,2, \ldots, n$.

Описанный метод отличается от большинства алгоритмов размещения чрезвычайной простотой. Он может быть использован даже при ручной методике разработки схем для грубого начального варианта размещения элементов. Опыт использования данного алгоритма показывает, что он, как правило, дает решение, несколько уступающее по качеству размещению, выполненному опытным конструктором, тогда как последовательные алгоритмы дают сокращение длины соединений на 10-15% [].

**Структура итерационных алгоритмов.** Алгоритмы данной группы используют общие идеи методов последовательных приближений и являются комбинаторными аналогами градиентных методов оптимизации. Для этих алгоритмов необходимо задать начальный вариант размещения. Итерационные алгоритмы применяются для решения задачи размещения с различными критериями оптимизации *F(p):* суммарная длина соединений, суммарное число пересечений соединений и т. д. В любом итерационном алгоритме исследуется некоторое подмножество размещений, в некотором смысле близких к начальному, для выделения в нем размещения с меньшим значением функции-критерия. Найденное размещение, вновь принимается за начальное, и процесс повторяется. Алгоритм завершается при отыскании некоторого размещения, в окрестности которого отсутствуют варианты с меньшим значением функции-критерия. В большинстве случаев такой процесс приводит к получению локального минимума функции *F(p).*

Пусть *F(p )*–некоторая функция-критерий размещения, а $p_{нач}$ – начальное размещение. Тогда в результате применения итерационного алгоритма размещения получается последовательность размещений $p_{нач}$,, $p_1$, $p_2$, …, $p_*$, которой соответствует монотонно убывающая последовательность значений/ $F(p_{нач})$,> $F(p_1)$> >$F(p_2)$ > ... >$F(p_*)$. *З*начение $F(p_*)$, вообще говоря, соответствует локальному минимуму функции. Однако практика применения подобных алгоритмов показывает, что получаемые размещения близки к оптимальным. Это, по-видимому, связано с тем, что используемые критерии оптимизации являются относительно пологими функциями, не имеющими «острых» экстремумов. Характерной чертой итерационного алгоритма является возможность получения варианта размещения в любой момент итерационного процесса. Поэтому при программной реализации алгоритма, как правило,

итерационный процесс заканчивается, как только разность значений функции-критерия для двух соседних итераций становится относительно малой:

$$[F(p_k) - F(p_{k-1})]/F(p_k) < \delta ,$$

где δ – заранее заданное число.

Различные итерационные алгоритмы размещения имеют сходную структуру, содержащую следующие элементы:

1) преобразование очередного размещения;

2) вычисление функции размещения;

3) выбор лучшего размещения на основе п. 2;

4) переход к следующей итерации и правило остановки.

В качестве начального может быть взято размещение, полученное одним из конструктивных алгоритмов размещения или заданное конструктором.

Назовем окрестностью очередного размещения множество вариантов, рассматриваемых при выборе лучшего размещения. Иногда это размещения, получаемые из данного попарными или групповыми перестановками элементов. В других случаях это размещения, получаемые при оптимизации мест расположения отдельных элементов, и т. д. В любом случае должно быть определено правило выбора размещения с меньшим значением *F(p).* При этом могут быть использованы различные тактики, например выбор на основе максимального уменьшения *F (p)* или выбор первого размещения с меньшим значением *F(p)* и т. д.

Отметим, что из-за отсутствия теоретических оценок эффективности различных итерационных алгоритмов невозможно отдать явное предпочтение одному из них.

**Алгоритмы парных перестановок.** Алгоритмы парных перестановок являются простейшими в рассматриваемом классе алгоритмов размещения.

Пусть имеется некоторое размещение (начальное или результат предыдущей итерации). Выбираются два элемента $e_i$ и $e_j$, и меняются местами. Рассчитывается новое значение *F(p);* если оно меньше прежнего, то произво-

дится обмен. Выбирается другая пара элементов и осуществляется аналогичная процедура. Процесс итеративно продолжается до тех пор, пока не будет применено используемое в алгоритме правило остановки.

Рассмотрим минимизацию суммарной длины соединений (1.15). Найдем приращение значения функции (1.15) при перестановке местами элементов $e_i$ и $e_j$ находящихся первоначально в позициях $h$ и $k$ соответственно.

На рисунке 1.7 показано положение этих элементов до и после перестановки, а также их связи с некоторым элементом $e_s(s\neq i\neq j)$, не участвующим в перестановке и находящимся в позиции $p(s)$.

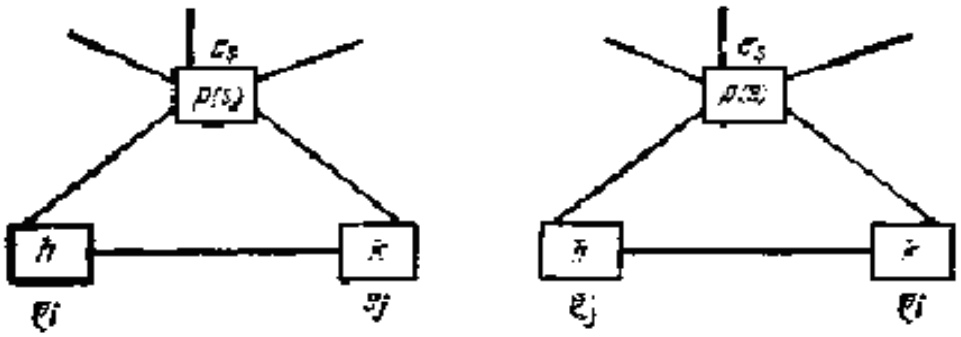


Рисунок 1.7 - Положение элементов до и после перестановки

Если длину соединений между всеми парами элементов, не затрагиваемых данной перестановкой, обозначить через $C$, то получим следующее выражение суммарной длины соединений:

$$F = \sum_s r_{is} d_{hp(s)} + r_{ij} d_{hk} + \sum_s r_{js} d_{kp(s)} + C \qquad (1.29)$$

После перестановки местами элементов $e_i$ и $e_j$ значение суммарной длины изменится и станет равным

$$F' = \sum_s r_{js} d_{hp(s)} + r_{ij} d_{hk} + \sum_s r_{is} d_{kp(s)} + C \qquad (1.30)$$

Сравнивая (1.29) и (1.30), легко рассчитать приращение функции $\Delta F_{ij}=F-F'$. После некоторых преобразований получим

$$\Delta F = \sum_s (r_{is} - r_{js})(d_{hp(s)} - d_{kp(s)}), \quad s \neq i, j \qquad (1.31)$$

Алгоритмы парных перестановок для минимизации суммарной длины соединений отличаются как способом определения исследуемой окрестности очередного варианта размещения, так и стратегией выбора следующего варианта с меньшим значением длины соединений. Очередная пара элементов для перестановки выбирается либо случайно, либо, что более эффективно, некоторым систематическим образом.

Можно например, вначале все элементы упорядочиваются в соответствии с убыванием характеристики

$$r_i = \sum_{j=1}^{n} r_{ij}, \; i = 1,2,...,n$$

которая рассчитывается по матрице соединений $R$ и определяет количество соединений элемента $e_i$ $c$ остальными. В результате находится последовательность номеров элементов:

$$I = i_1, i_2, ..., i_n \;\; (r_{i_1} \geq r_{i_2} \geq ... \geq r_{i_n})$$

На очередной итерации для каждого элемента $i_k$ в последовательности $I$ рассчитывается приращение $\Delta F$ (4.82) при условии перестановки этого элемента с элементами, стоящими правее в последовательности $I$. Далее выбирается максимальное положительное значение $\Delta F$; и элемент $i_k$ меняется местами с соответствующим элементом последовательности. Если для данного

элемента $i_k$ в наборе $\{ \Delta F_{i_k j} \}$ отсутствуют положительные приращения, он оставляется на своем месте и осуществляется переход к отысканию «наилучшего» места для следующего элемента *ik+1.* Таким образом, для определения места элемента $i_1$ рассчитывается *n-1* приращений, элемента $i_2$ рассчитывается *n-2* приращений и т. д. Всего на одной итерации рассчитывается *n(n-1)/2* приращений и уточняется местоположение всех *n* элементов.

По окончании очередной итерации получается размещение с меньшей длиной соединений и осуществляется переход к следующей итерации.

Итерационный процесс заканчивается, когда изменение общей длины соединений в соответствии с (1.32) становится относительно малым.

Можно модифицировать описанный алгоритм, учитывая изменение длин соединений отдельных элементов при перестановках. Последовательность выбора элементов для перестановки определяется не количеством соединений $r_i$ , а их суммарной длиной:

$$F_i = \sum_{j=1}^{n} r_{ij} d_{p(i)p(j)}, \quad i = 1, 2, ..., n$$

При этом в первую очередь уточняется расположение элементов с большими значениями $F_i$. После каждого успешного обмена значения $F_i$ пересчитываются, и процесс повторяется.

Методы решения специальных задач размещения. В рамках решения специальных задач размещения учитывается не только оптимизация целевой функции (1.15), но одновременно с этим дополнительных критериев оптимальности, в чем и состоит преимущество перед другими отдельно взятыми алгоритмами и методами.

Максимальная длина соединений в схеме непосредственно определяет задержку при распространении электрического сигнала. В связи с этим при конструировании определенного класса схем минимизация максимальной

длины соединений может быть более важным критерием, чем МСД соединений. Отметим, что критерий минимизации СДС обеспечивает лишь в среднем малую длину отдельных соединений, возможность минимизации максимальной длины соединений возможна и косвенным образом в процессе минимизации суммарной длины соединений.

Теперь же остановимся на прямых методах уменьшения длин отдельных соединений. Для математической постановки задачи введем матрицу C=||cij||nxn в которой cij=l, если элементы ei и ej связаны (rij>0), cij=0 в противном случае.

Задачу размещения можно сформулировать следующим образом:

найти перестановочную матрицу X0 порядка n, для которой

$$F(X_0) = \min_{\{X\}} \max_{\substack{1 \le i,\ j \le n \\ 1 < k,\ s \le n}} c_{ij} d_{ks} x_{ik} x_{js}, \tag{1.32}$$

где {X}— множество перестановочных матриц порядка n(|{X}|=n!).

Задачей на узкие места при назначении называется задача определения перестановочной матрицы $X_0$= || $x_{ij}$||nxn для которой

$$F(X_0) = \min_{\{X\}} \max_{1 \le i, i \le n} a_{ij} x_{ij}, \tag{1.33}$$

Имея в виду (1.32), в отличие от задачи квадратичного назначения, задача (1.33) может быть названа задачей на узкие места при «квадратичном назначении. В настоящее время неизвестны какие-либо, отличные от полного перебора, алгоритмы решения задачи (1.32). В то же время возможны квазиоптимальные методы ее решения. Рассмотрим основные идеи алгоритма А. Я.Чумакова, предназначенного для улучшения некоторого начального варианта размещения.

Для определенности будем считать, что каждый источник сигнала соединен с его приемниками по схеме типа «звезда», а расстояния между позициями рассчитываются по $d_{ij}=|x_i-x_j|+|y_i-y_j|$. Для улучшения начального варианта размещений" используется метод парных перестановок, по

структуре аналогичный алгоритму минимизации максимального числа выводов на узлах.

1) Найти элемент $e_i$ со связью максимальной длины.
2) Найти другой элемент $e_j$ такой, что перестановка пары ($e_i$ , $e_j$) уменьшает число связей максимальной длины и не приводит к образованию еще более длинных связей.
3) Из всех $e_j$ отобранных по шаг2, выбрать такой элемент е., что перестановка пары ($e_i$ , $e_{j*}$) ликвидирует наибольшее число связей максимальной длины; произвести перестановку ($e_i$ , $e_{j*}$ ).
4) Повторять шаги 1—3до тех пор, пока либо не исчезнут все связи максимальной длины, либо ни одной из допустимых перестановок не удается уменьшить число связей максимальной длины.
5) В первом случае перейти к уменьшению количества следующих по длине связей (шаги 1—3), во втором — процесс улучшения заканчивается.

Экспериментальная проверка алгоритма на ЭВМ по казала, что максимальная длина соединений при машин ном размещении составляла порядка 75—80% от соответствующей величины при размещении, выполненном конструктором.

Используя алгоритм размещения «минимизация наидлиннейших связи», достигается одновременное улучшение таких показателей как время прохождения электрического сигнала в схеме электронного устройства (за счёт минимизации наидлиннейших связей), а так же, из-за наличия основного главного критерия – целевой функции(СДС) происходит минимизация суммы длин всех соединений тем самым обеспечивается улучшение общих электрических характеристик схемы.

### 1.3 Эффективность алгоритмов размещения

Алгоритм - это последовательность четко определенных инструкций, предназначенных для решения некоторой задачи. Другими словами, это последовательность команд, позволяющих получить из корректных входных данных требующиеся входные данные за ограниченный промежуток времени.

Понятие алгоритма включает в себя важные моменты:

— Каждый шаг алгоритма должен быть четко и однозначно определен. Это требование является обязательным и не должно нарушатся ни при каких обстоятельствах.

— Должны быть точно указаны диапазоны допустимых значений входных данных, которые обрабатываются с помощью алгоритма.

— Один и тот же алгоритм можно представить несколькими разными способами.

— Для решения одной и той же задачи может существовать несколько разных алгоритмов.

— В основу алгоритмов для решения одной и той же задачи могут быть положены совершенно разные принципы, что может существенно повлиять на скорость решения этой задачи.

Существует два вида оценки эффективности алгоритма: временная и пространственная. Временная эффективность является индикатором скорости работы алгоритма. Что касается пространственной эффективности, то эта оценка показывает количество дополнительной оперативной памяти, необходимой для работы алгоритма.

Еще одной важной характеристикой алгоритма является его простота. Простые алгоритмы легче понять и запрограммировать. Следовательно, в полученной программе будет содержаться меньше ошибок.

Следующей важной характеристикой алгоритма является его общность, или универсальность. По сути, здесь можно выделить два момента: общность

задачи, для решения которой разработан алгоритм, и диапазон допустимых значений его входных данных. По поводу первого замечания следует сказать, что иногда легче разработать алгоритм для решения общей задачи, чем заниматься поиском решения ее частного случая.

Что касается диапазона допустимых значений входных данных алгоритма, то здесь нужно отметить следующее. При разработке алгоритма нужно учитывать, что диапазон изменения значений входных параметров может колебаться в очень широких пределах и должен естественным образом соответствовать решаемой задаче.

Алгоритмы, как и аппаратное обеспечение компьютера, следует рассматривать как технологию. Общая производительность системы настолько же зависит от эффективности алгоритма, насколько и от мощности применяющегося аппаратного обеспечения.

Различные алгоритмы, разработанные для решения одной и той же задачи, часто очень сильно различаются по эффективности. Эти различия могут быть намного значительнее тех, которые вызваны применением неодинакового аппаратного и программного обеспечения.

Все алгоритмы разделяют на такие, которым достаточно ограниченной памяти, и те, которым нужно дополнительное пространство. Нередко программистам приходилось выбрать более медленный алгоритм просто потому, что он обходился имеющейся памятью и не требовал внешних устройств.

Время выполнения большинства алгоритмов напрямую зависит от размера вводимых данных (т.е. чем больше размер, тем дольше работает алгоритм). В некоторых алгоритмах для оценки размерности входных данных может использоваться сразу несколько параметров (например, количество вершин и ребер для тех алгоритмов, в которых граф представляется в виде связанных списков смежных вершин).

Выбор параметра, показывающего размер входных данных, имеет особое значение. Один из примеров подобных случаев - перемножение двух матриц

размером $n \times n$. Существует две подходящих единицы измерения размера данной задачи. первая и наиболее часто используемая - это порядок матрицы $n$.Однако существует и другая, не менее подходящая единица измерения, количество $N$ перемножаемых элементов в матрицах. Последняя единица к тому же более универсальна, поскольку ее можно применять не только к квадратным матрицам. Поскольку существует простая формула, связывающая эти две единицы измерения, мы легко можем перейти от одной единицы к другой, однако искомая эффективность алгоритма будет в значительной степени отличатся в зависимости от того, какая из двух единиц была использована при решении задачи.

На выбор подходящей системы измерений размера задачи могут повлиять выполняемые рассматриваемым алгоритмом действия.

Мы должны рассмотреть еще один вопрос, касающийся единиц измерения времени выполнения алгоритма. Безусловно, для этой цели можно просто воспользоваться общепринятыми единицами измерения времени - секундой, миллисекундой и т.д. и с их помощью оценить время выполнения программы, реализующей рассматриваемый алгоритм. Однако у такого подхода существуют явные недостатки, поскольку результаты измерений будут зависеть от:

— быстродействия конкретного компьютера;

— тщательности реализации алгоритма в виде программы;

— типа компилятор, использованного для генерации машинного кода;

— точности хронометрирования реального времени выполнения программы.

Поскольку перед нами стоит задача измерения эффективности алгоритма, а не реализующей его программы, мы должны воспользоваться такой системой измерений, которая бы не завесила от приведенных выше посторонних факторов.

Один из возможных способов решения этой проблемы состоит в подсчете того, сколько раз выполняется каждая операция алгоритма. Однако подобный

подход слишком сложен и, как мы увидим в дальнейшем, чаще всего не нужен. Поэтому мы должны составить список наиболее важных операций, выполняемых в алгоритме, называемых основными, или базовыми операциями, определить, какие из них вносят наибольший вклад в общее время выполнения алгоритма, и вычислить, сколько раз эти операции выполняются.

Почему выше мы сделали замечание по поводу вычисления порядка роста количества основных операций алгоритма для достаточно больших размеров входных данных? Дело в том, что при малых размерах входных данных невозможно заметать разницу во времени выполнения между эффективным и неэффективными алгоритмом.

Однако существует большое количество алгоритмов, время выполнения которых зависит не только от размера входных данных, но также и от особенностей конкретных входных данных. Время работы алгоритма может отличатся в очень широких пределах для одного и того же списка размера *n.*

Существуют понятия наихудшего и наилучшего случая. Под эффективностью алгоритма в наихудшем случае подразумевают его эффективность для наихудшей совокупности входных данных размером *n*, т.е. для такой совокупности входных данных размером *n* среди всех возможных, для которой время работы алгоритма будет наибольшим.

Под эффективностью алгоритма в наилучшем случае подразумевают его эффективность для наилучшей совокупности входных данных размером *n*, т.е. для такой совокупности входных данных размером *n* среди всех возможных, для которой время работы алгоритма будет наименьшим.

На основании информации, полученной в результате анализа алгоритма для наилучшего и наихудшего случаев, нельзя сделать вывод о том, как поведет себя алгоритм при обработке типовых или случайно заданных входных данных. Чтобы получить подобную информацию, нужно выполнить анализ алгоритма для среднего случая.

Существует еще один вид эффективности, называемый амортизированной эффективностью. Она предназначена не столько для анализа

общего времени выполнения алгоритма, сколько для анализа последовательности операций, выполняемых над одной и той же структурой данных. Бывает так, что одна из операций алгоритма может оказаться слишком продолжительной, но общее время выполнения последовательности из *n* таких операций будет значительно меньше, чем его оценка, выполненная для наихудшего случая, равная произведению максимального времени выполнения этой операции на число таких операций, n. Поэтому мы можем разделить (или "амортизировать") высокую стоимость выполнения алгоритма для наихудшего случая на всю последовательность выполняемых операций по аналогии с тем, как в экономике разносят стоимость дорогостоящего оборудования на весь период его эксплуатации.

Скорость роста алгоритма играет важную роль, которая определяется старшим, доминирующим членом формулы. Отбросив все младшие члены, мы получаем то, что называется порядком функции или алгоритма, скоростью роста их сложности которого она является. Алгоритмы можно сгруппировать по скорости роста их сложности. Введем три категории: алгоритмы, сложность которых растет по крайней мере также быстро, как данная функция, алгоритмы, сложность которых растет с той же скоростью, и алгоритмы, сложность которых растет медленнее, чем эта функция.

Для того чтобы можно было сравнивать между собой эти порядки роста и классифицировать их, ученые ввели три условных обозначения: *O* (прописная "O"), Θ (прописная греческая "тета") и Ω (приписная греческая "омега").

**Омега большое**. Класс функций, растущих по крайне мере также так же быстро, как /, мы обозначаем через Ω(f) (читается омега большое). Функция δ принадлежит этому классу, если при всех значениях аргумента *n*, больших некото*рого порога по, значение* $\delta(n) > f(n)$ для некоторого положительного числа *c*. Можно считать, что класс Ω(f) задается указанием свой нижней границы: все функции из него по крайней мере так же быстро, как /.

Мы занимаемся эффективность алгоритмов, поэтому класс Ω(f) не будет представлять для нас большого интереса: например, в $\Omega(n^2)$ входят все функции, растущие быстрее, чем $n^2$, скажем $n^3$ и 2".

**О большое**. На другом конце спектра находится класс O(f) (читается *о* большое). Этот класс состоит из функций, растущих не быстрее /. Функция / образует верхнюю границу для класса O(f). С формальной точки зрения функция δ принадлежит классу O(f), если $g(n) < cf(n)$ для всех $n$, больших некоторого порога по, и для некоторой положительной константы $c$.

Этот класс чрезвычайно важен для нас. Про два алгоритма нас будет интересовать, принадлежит ли сложность первого из них классу О большое от сложности второго. Если это так, то, значит, второй алгоритм не лучше первого решает поставленную задачу.

Тета большое. Через Θ(f) (читается тета большое) мы обозначим класс функций, растущих с той же скоростью, что и /. С формальной точки зрения этот класс представляет собой пересечение двух предыдущих классов, Θ(f)=Ω(f)⋂O(f).

При сравнении алгоритмов нас будут интересовать такие, которые решают задачу быстрее, чем изученные. Поэтому, если найденный алгоритм относится к классу в, то он нас не очень интересен. Мы не будем часто ссылается на этот класс.

При изучении вычислительной сложности задач первое, на что смотрят и ученые, и практики в области информатики, - может ли данная задача быть разрешена при помощи некоторого алгоритма за полиноминальное время.

Алгоритм решает задачу за полиноминальное время, если его временная эффективность в наихудшем случае принадлежит классу $O(p(n))$, где $p(n)$ - полином от размера в входных данных $n$. Задачу, которая может быть рушена за полиноминальная время, будем называть ***легкой,*** а задачу, которая не может быть решена за полиноминальное время, - ***трудной***.

Имеется ряд причин для определения трудности задач таким образом.

Во-первых, хотя и разумно считать трудноразрешимой задачу, для решения которой требуется время $\Theta(n^{100})$, на практике крайне редко встречаются задачи, время решения которых выражается полиномом такой высокой степени. Для практических задач, которые решаются за полиномиальное время, показатель степени обычно намного меньше. Опыт показывает, что если для задачи становится известен алгоритм с полиномиальным временем работы, то зачастую впоследствии разрабатывается и более эффективный алгоритм. Даже если самый лучшей из известных на сегодняшний день алгоритмов решения задачи характеризуется временем $\Theta(n^{100})$, достаточно высока вероятность того, что вскоре будет разработан алгоритма с намного лучшим временем работы.

Во-вторых, для многих приемлемых вычислительных моделей задача, которая решается в течение полиномиальное время и в другой. Например, в большой части книги [1] рассматривается класс задач, разрешимых в течение полиномиального времени с помощью последовательных машин с произвольным доступом к памяти. Этот класс совпадает с классом задач, разрешимых в течение полиномиального времени на абстрактных машинах Тьюринга. Он также совпадает с классом задач, разрешимых в течение полиномиального времени на параллельных компьютерах, если зависимость количества процессоров от объема входных данных описывается полиномиальной функцией.

В-третьих, класс задач, разрешимых в течение полиномиального времени, обладает полезными свойствами замкнутости, поскольку множество полиномов замкнуто относительно операций сложения, умножения и композиции. Например, если выход одного алгоритма с полиномиальным составной алгоритм.

Один из способов решения задач состоит в том, чтобы свести, или редуцировать, одну задачу к другой. Тогда алгоритма решения второй задачи можно преобразовать таким образом, чтобы он решал первую. Если преобразование выполняется за полиномиальное время и вторая задача

решается за полиномиальное время, то и наша новая задача также решается за полиномиальное время.

Задача принятия решения D является ***NP-полной*** если:

1) она принадлежит классу *NP*;
2) любая задача в NP полиноминально приводима к D.

Показать, что задача принятия решения является *NP*-полной, можно в два этапа. Сначала надо показать, что рассматриваемая задача является *NP*-задачей, т.е. что случайным образом сгенерированная строка может быть проверена на пригодность в качестве решения задачи за полиномиальное время. Обычно этот этап весьма прост. Второй этап заключается в том, чтобы показать, что любая задача из множества *NP* приводима к рассматриваемой задаче за полиномиальное время. Благодаря транзитивность полиномиального приведения для выполнения этого этапа достаточно показать, что известия *NP*-полная задача может быть приведена к данной за полиномиальное время.

Предположим, мы работаем над задачей оптимизации, каждому из возможных решений которой сопоставляется положительная стоимость, и требуется найти решение, близкое к оптимальному. В зависимости от задачи оптимальное решение можно определить либо как такое, которому соответствует минимально возможная стоимость; другими словами, это может быть либо задача максимизации, либо задача минимизации.

Алгоритм, в котором достигается коэффициент аппроксимации *p(n),* будем называть *p(n)*-приближенным алгоритмом.

Говорят, что алгоритм решения задачи обладает отношением, или коэффициентом, аппроксимации *p(n)*, если для произвольных входных данных размером *n* стоимость *C* решения, полученного в результате выполнения этого алгоритма, отличается от стоимости *C** оптимального решения не более чем в *p(n)* раз:

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq p(n) \qquad (1.34)$$

деления коэффициента аппроксимации и *p(n)*-приближенного алгоритма применимы и к задачам минимизации, и к задачам максимизации. Для задач максимизации выполняется неравенство $0 < C \leq C^*$, и отношение $C^*/C$ равно величине, на которую стоимость оптимального решения больше стоимости приближенного решения. Аналогично для задач минимизации выполняется неравенство $0 < C \leq C^*$, и отношение $C^*/C$ дает ответ, во сколько раз стоимость приближенного решения больше стоимости оптимального решения. Поскольку предполагается, что стоимости всех решений положительные, эти коэффициенты вполне определены. Коэффициент аппроксимации приближенного алгоритма не может быть меньше 1, поскольку из неравенства $C/C^* < 1$ следует неравенство $C^*/C > 1$. Таким образом, 1-приближенный алгоритм выдает оптимальное решение, а приближенный алгоритма с большим отношением аппроксимации может возвратить решение, которое намного хуже оптимального.

Для многих задач разработаны приближенные алгоритмы с полиномиальным временем работы и малыми постоянными отношениями аппроксимации. Есть также задачи, для которых лучше из известных приближенных алгоритмов с полиномиальным временем работы характеризуются коэффициентами аппроксимации, величина которых возврастает с ростом размера входных данных *n*.

Некоторые *NP*-полные задачи допускают наличие приближенных алгоритмов с полиномиальным временем работы, коэффициент аппроксимации которых можно уменьшать за счет увеличения времени из работы. Другими словами, в них допускается компромисс между временем вычисления и качеством приближения.

Схема аппроксимации задачи оптимизации - это приближенный алгоритм, входные данные которого включается в себя не только параметры экземпляра задачи, но и такое значение $\varepsilon > 0$, что для любого фиксированного значения $\varepsilon$ *это схема является* $(1+\varepsilon)$-приближенным алгоритмом. Схему аппроксимации называются схемой аппроксимации с полиномиальным

временем выполнения, если для любого фиксированного значения $\varepsilon>0$ работа этой схемы завершается за время, выраженное полиномиальной функцией от размера $n$ входных данных.

Время работы схемы аппроксимации с полиномиальным временем вычисления может очень быстро возрастать при уменьшении величины $\varepsilon$. Например это время может вести себя как $O(n^{2/\varepsilon})$. В идеале, если величина $\varepsilon$ уменьшается на постоянный множитель, время, необходимо для достижения нужного приближения, не должно возрастать более чем на постоянный множитель.

Говорят, что схема аппроксимации является схемой аппроксимации с полностью полиномиальным временем работы, если время работы выражается полиномом как от $1/\varepsilon$, так и от размера входных данных задачи $n$. Например, время работы такой схемы может вести себя как $O((1/\varepsilon)^2 n^3)$ . В такой схеме любое уменьшение величины $\varepsilon$ на постоянный множитель сопровождается соответствующим увеличением времени работы на постоянный множитель.

Объективная оценка эффективности алгоритмов размещения должна опираться на анализ точности решений и времени их получения в зависимости от основных параметров задачи. Однако проведение такого анализа в общем виде практически невозможно. Наиболее реально получение таких оценок при применении алгоритмов к задачам определённого класса. В настоящее время имеется ограниченное число тестовых задач, по которым известны оценки эффективности различных алгоритмов размещения. Одной из них является тест-задача Штейнберга [3], а также применяют статистические методы [3]. Производится обработка результатов решения задачи размещения различной размерности со схемами соединений, получаемыми с помощью специальных программ генераторов. При проведении таких оценивается среднее время решения, среднее значение целевой функции и ее изменение. Для сравнения качества решений, получаемых различными алгоритмами, часто пользуются статистическим критерием знаков [3].

Недостаточность экспериментальных данных не позволяет сделать окончательный вывод об эффективности алгоритмов. При интерпретации экспериментальных результатов, полученных с помощью статистических методов оценки эффективности, необходимо принимать во внимание параметры исследованных задач (структуру схем соединений, размерность задач и т.д.). Как правило, при таких сравнениях размеры задач относительно невелики, а структура случайных схем соединений не адекватна реальным схемам, поэтому алгоритмы наиболее эффективные для одного класса задач, задач перестают быть таковыми для реальных задач размещения. В связи с этим задачей дальнейших исследований является проверка эффективности алгоритмов на схемах, параметры которых близки которых близки к параметрам реальных схем соединений.

Время работы алгоритмов обычно бывает сложной функцией параметров задачи и в общем случае даже порядок роста вычислений при варьировании параметров неизвестен. Иногда порядок роста может быть грубо оценен как функция от числа размещаемых элементов. Некоторые алгоритмы начального размещения и перестановочные алгоритмы требуют около $n^2$ вычислений, в то же время имеются последовательные матричные алгоритмы со степенью роста вычислений $n^4$ и даже $n^5$. Естественно, что эти оценки говорят лишь о тенденциях роста вычислений, а не о действительных затратах времени при данном числе элементов n.

На основании значительного практического опыта и исследований, выполненных различными авторами, можно дать лишь общие рекомендации по применению алгоритмов размещения.

Возможность выбора алгоритма для решения задачи размещения предоставят результаты тестирования алгоритмов на устройствах заданного класса. Наличие механизма выбора алгоритма размещения значительно увеличит качество автоматизации процесса решения задачи размещения.

## Выводы  по главе 1

Увеличения качества автоматизации решения задачи размещения за счёт механизма выбора наиболее эффективного алгоритма размещения может быть достигнуто при наличии информации о эффективности алгоритмов размещения для устройств заданного класса, которую возможно получить только при условии наличия генератора коммутационных схем.

## 2 ГКС как основа оценки эффективности алгоритмов размещения

Для выявления эффективности алгоритмов размещения необходимо выполнить следующие условия:

1) Определить элементную базу для которой нужно провести тестирование алгоритмов;
2) Иметь в наличии коммутационные схемы соответствующие заданной элементной базе;
3) Количество коммутационных схем должно быть не меньше 1000.

Современный подход выявления эффективности алгоритмов размещения имеет низкую степень доверия. Создатели алгоритмов размещения утверждают что результаты работы создаваемых алгоритмов более качественны нежели существующие алгоритмы размещения. Однако авторы алгоритмов не предоставляют результаты вычислительных экспериментов либо тестовые данные описаны не достаточно подробно [8-17], либо количество тестовых данных недостаточное для объективной оценки. А также не учитывается элементная база, для которой выполняется задача размещения. Зачастую не выполняются 2 из 3 условий проведения тестирования алгоритмов.

### 2.1 Выявление параметров генерации

Разнообразия элементной база электронных устройств для алгоритмов размещения проявляется в основных параметрах, описывающих коммутационную схему, являются количество элементов (КЭ) и характер связности элементов. Под характером связности подразумевается, каким образом элементы соединены между собой. Каждый элемент имеет связь с одним и более элементов, и с каждым из них имеет от одного и более соединений. Далее введём термины, которые будут использованы для описания характера связности: факт связи (ФС) – это наличие связи между элементами; количество связей (КС) – это количество соединений элемента.

Для выявления параметров генерации необходимо выбрать наиболее подходящие параметры. Параметры которые могут быть основой для генерации представлены в таблице 2.1.

Таблица 2.1 – Параметры коммутационной схемы

| **Параметры элемента** | **Параметры схемы** |
|---|---|
| Список всех элементов с которыми имеется факт связи | Количество элементов |
| Количество ФС | Количество ФС (без повторений) |
| КС | КС (без повторений) |
| Список содержащий КС с каждым элементов | Отношение используемых ФС к максимальному ФС схемы |
| | Среднее количество ФС на элемент |
| | Среднее КС на элемент |
| | Максимальное ФС элемента |
| | Максимальное КС элемента |
| | Список, каждый элемент которого содержит количество одинаковых вариантов ФС |

Все параметры схемы можно разделить на категории: параметры всей схемы, параметры элемента, параметры связи между элементами. Параметр связи включает в себя два элемента между которыми присутствует связь т.е. ФС, КС между этими элементами. Параметры элемента включают в себя множество связей, в которых участвует элемент, множество элементов с которыми имеется ФС, сумма КС со всеми элементами. В параметры схемы входит множество элементов и множество связей.

Для более детального описания схемы нужно соотношение элементов, имеющих различные параметры ФС и КС, то есть численное наименования элементов с любым возможным сочетанием значений параметров ФС и КС.

Все элементы имеют параметры ФС и КС, одинаковые значения этих параметров встречаются с различной периодичностью. При подсчёте количества элементов с различными показателями для каждого параметра можно получить общие представление о соотношении элементов в схеме. Для визуализации этой информации удобно использовать гистограммы.

Для построения и отрисовки гистограмм разработана вспомогательная программа анализатор, которая выявляет показатели параметров схемы.

Программа выявляет около 30 параметров схемы, на основании которых нужно определить параметры генерации схемы.

На рисунке 2.1 приведена гистограмма ФС, отображающая количество элементов с одинаковым показателем ФС.

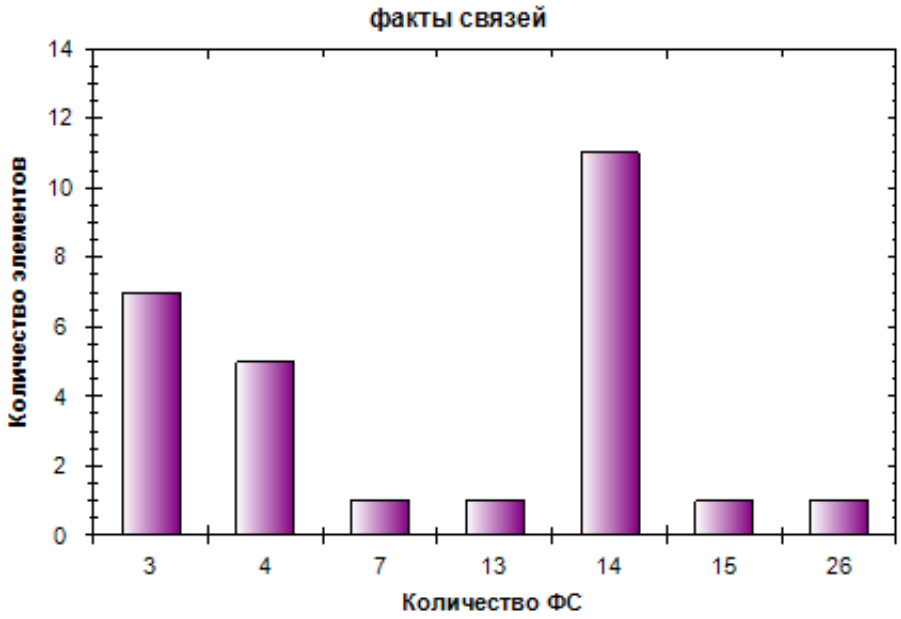



Рисунок 2.1 – Гистограмма ФС

Гистограммы ФС и КС описывают схему подробно, являются продуктом анализа схемы, используя только их можно создать схему. То есть генерация гистограмм позволяет частично описать схему, а именно указать количество групп элементов с разным значением параметра и количество элементов в каждой группе. Для генерации гистограмм нужны параметры содержащие информацию о характере распределения элементов: количество элементов, ФС или КС в зависимости от гистограммы, границы гистограммы и какое либо вероятностное распределение.

По виду гистограмм можно сказать, что они частично, либо полностью соответствуют какому-либо вероятностному распределению. Для большинства схем можно выделить участки с нормальным распределением исходных параметров, в частности, ФС.

На рисунке 2.2 приведена гистограмма ФС, на диапазоне от нуля до десяти имеется распределение количества элементов близкое к нормальному.

На одной гистограмме может быть несколько диапазонов, имеющих различные виды распределений. Разделение гистограммы на диапазоны, имеющие различное распределение, предоставляет возможность описать гистограмму более подробно. Генерация каждого диапазона в отдельности позволит получить всю гистограмму параметров, в частности, ФС. При увеличении числа диапазонов возрастает точность описания гистограммы, следовательно и точность описания схемы.

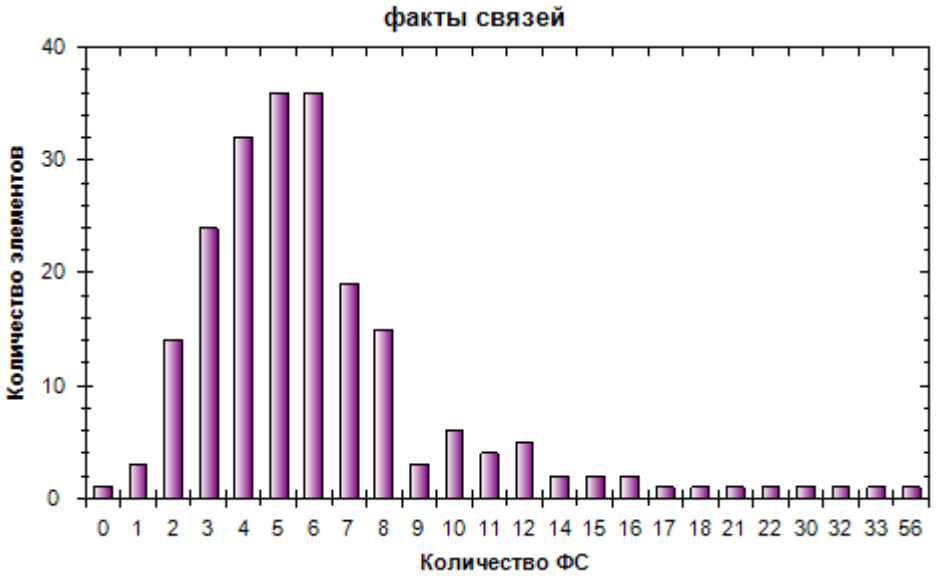



Рисунок 2.2 – Гистограмма ФС с фрагментом нормального распределения

Гистограмму КС возможно применить при генерации схемы, однако она не указывает какие именно элементы имеют определённое значение КС, из-за этого проблематично конкретизировать, какое количество связей (КС) у каждого элемента. Решением является иное представление гистограммы КС, которое заключается в делении каждого диапазона ФС на диапазоны, в которых указывается КС и вид вероятностного распределения. В результате имеем структуру данных, позволяющую подробно описать распределение КС в схеме.

## 2.2 Формирование исходных данных

Во избежание ситуаций некорректной установки параметров требуются механизмы, гарантирующие соблюдение последовательности установки параметров и установку параметров в пределах разрешённых границ.

Параметры для генерации связаны между собой, изменение некоторого параметра влечёт за собой изменение зависящих от него параметров и наоборот. Например, предварительная установка КЭ ограничит параметр ФС, что повлияет на крайние границы гистограммы и КС, но если предварительно установить ФС, появится ограничение на КЭ. Установление одностороннего влияния позволит выявить последовательность установки параметров, сформировать этапы установки параметров, на каждом из которых определяются границы значения параметра на основании ранее заданных.

На рисунке 2.3 представлена схема взаимодействия параметров при одностороннем влиянии.

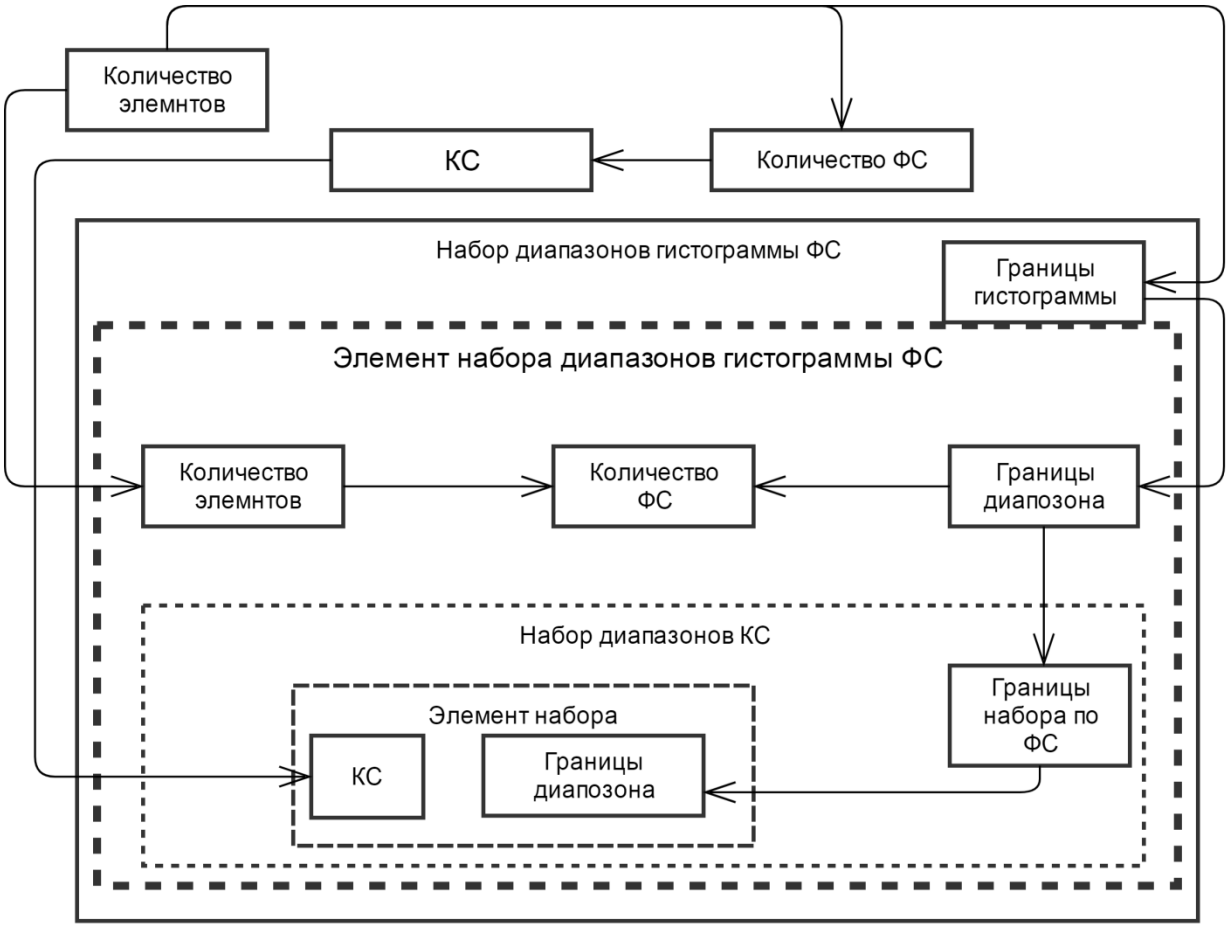



Рисунок 2.3 – Схема взаимодействия параметров при одностороннем влиянии

Порядок установки параметров при формировании данных для генератора коммутационных схем должен быть следующим:

1) Количество элементов;
2) Общий ФС в схеме;
3) Максимальный ФС у элемента (т.е. правая граница наборов ФС);
4) Диапазоны ФС;
5) Количество элементов в каждом диапазоне;
6) ФС в каждом диапазоне;
7) КС диапазонов ФС;
8) Диапазоны КС.

На рисунке 2.4 приведена схема влияния параметров генерации на границы их диапазонов с учётом одностороннего влияния.

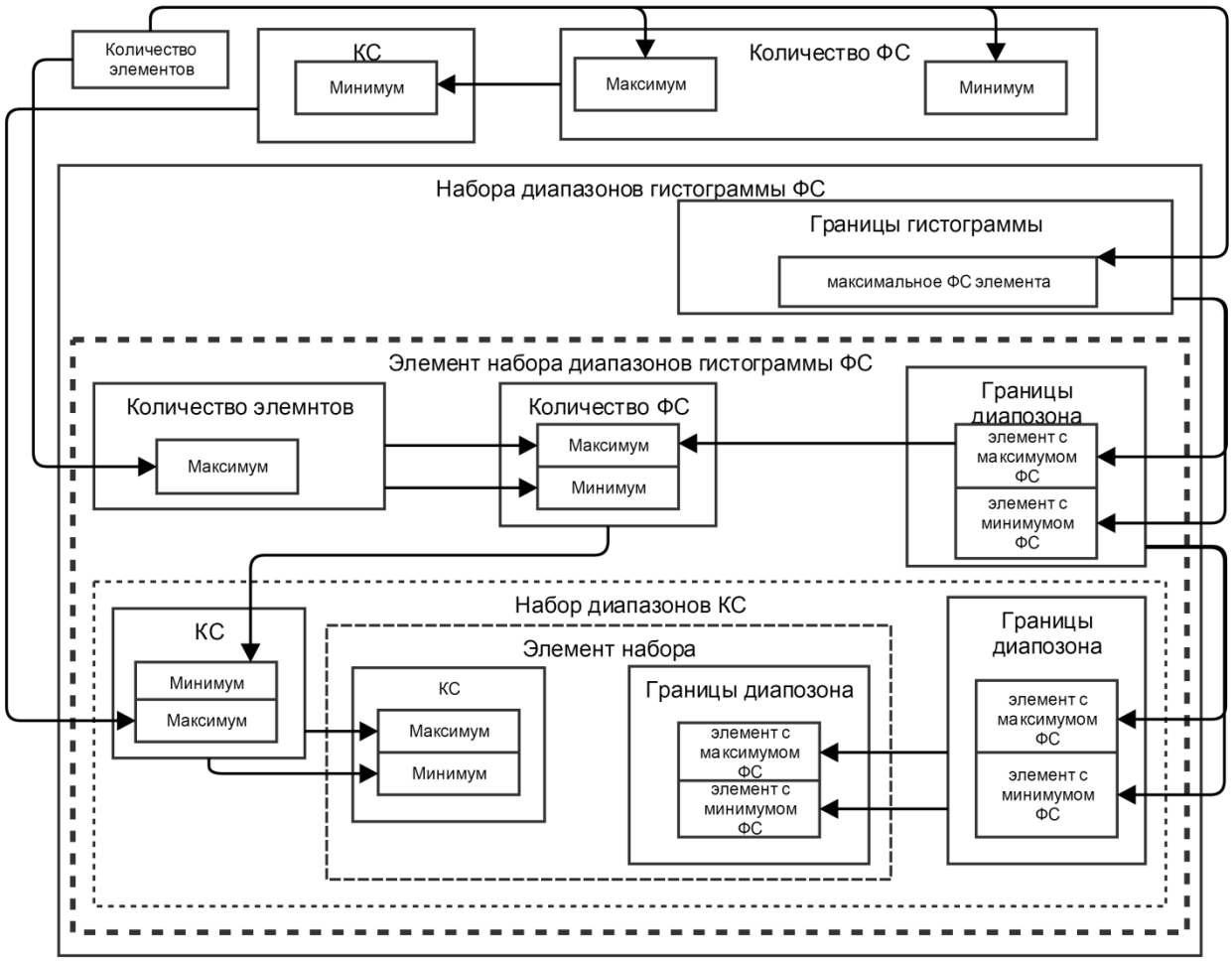



Рисунок 2.4 – Схема определения границ параметров

Алгоритм формирования данных для генерации коммутационных схем на основе объединения порядка установки параметров и их границ будет выглядеть следующим образом:

— Шаг 1 - задаётся количество элементов (параметр не имеет ограничений);

— Шаг 2- задаётся ФС, предварительно определяется возможный максимум и минимум. Минимум равен количеству элементов. Для вычисления максимума используется следующие выражение

$$\sum_{i=1}^{\mathrm{n}} i = (\mathrm{n} * (\mathrm{n} - 1))/\,2, \qquad (2.1)$$

где n- количество элементов;

— Шаг 3 - задаётся максимальный ФС у элемента, максимальное значение параметра определяется формулой

$$\Phi\mathrm{C}max = \mathrm{КЭ} - 1 \qquad (2.2)$$

— Шаг 4 - задаются диапазоны ФС, левая граница равна двум, так как при меньшем количестве ФС элемент служит внешним выводом устройства и принадлежит фиктивному элементу [3], а правая граница определяется на третьем шаге;

— Шаг 5 - каждому диапазону ФС задаётся количество элементов с условием, что сумма элементов всех диапазонов равна заданному на первом шаге количеству элементов;

— Шаг 6 - каждому диапазону задаётся ФС. При этом каждый диапазон имеет свои ограничения для ФС, которые определяются предварительно. Минимум и максимум ФС диапазона вычисляется по следующему алгоритму:

если ФС > КЭ +1, то используется формула:

$$\sum_{i=\Phi\mathrm{C}-\mathrm{КЭ}}^{\Phi\mathrm{C}} i \qquad (2.3)$$

Если ФС ≤ КЭ +1, то используется формула:

$$КЭ * ФС / 2 \quad (2.4)$$

Корректность определяется следующим образом, сумма ФС диапазонов должна равняться ФС заданному на втором шаге умноженному на два.

$$\sum_{i=0}^{n} ФС_i = ФС * 2, \quad (2.5)$$

где $i$ номер диапазона ФС;

— Шаг 7 - задаётся вид распределения диапазонов ФС (ограничения отсутствуют);

— Шаг 8 - для каждого диапазона ФС задается КС, минимум КС для диапазона это указанный ранее ФС этого диапазона;

— Шаг 9 - для каждого диапазона ФС задается набор диапазонов КС, минимально количество диапазонов КС это один. Для каждого диапазона КС указывается вид распределения и количество КС, минимум которого равен ФС текущего диапазона. Проверяется корректность КС по формуле

$$\sum_{i=0}^{n} КС_i = КС * 2, \quad (2.6)$$

где $i$ -номер диапазона ФС.

### 2.3 Принцип уточнения

Описанный выше алгоритм установки параметров позволяет контролировать степень подробности входных данных для генерации. Это осуществляется за счёт установки количества диапазонов ФС и КС. В соответствии с желаемой элементной базой устанавливается необходимое количество диапазонов, количество элементов для диапазонов и количество ФС.

Изначально имеется один диапазон с установленными границами ФС. При наличии только одного диапазона возможно создать схему, однако вероятно не совпадение с желаемой элементной базой пользователя. Для достижение приближенности к желаемой элементной базе пользователь создает несколько диапазон. Для этого создан механизм последовательного уточнения, не позволяющий пользователю указать не корректный результат.

Суть механизма заключается в делении существующего диапазона на две части по устанавливаемой пользователем границе. На практике это работает следующим образом:

1) Выбирается диапазон, который по мнению пользователя следует уточнить;
2) Указывается граница разделения;
3) Диапазон делится на две части.

Таким образом предполагается описание желаемой элементной базы.

### 2.4 Алгоритм генерации

Ключевая роль алгоритма генерации, это на основании имеющихся параметров генерации создать схему. Задача алгоритма генерации интерпретировать (дополнить), данные генерации, которые описывают схему приближенно, до полноценной коммутационной схемы.

Возможно различно интерпретировать параметры генерации. Алгоритм генерации может руководствоваться различными приоритетами при создании схемы, например:

— Каждый элементы из диапазона должен иметь минимум связей внутри диапазона;
— КС между соседними диапазонами должны быть больше, чем с не соседними;
— Разброс ФС элементов внутри каждого диапазона должен учитывать какое либо вероятностное распределение.

Для устранения неоднозначности конечного результата нужно либо добавить параметры генерации, либо иметь разнообразие алгоритмов генерации либо различные требования к генерации должны выступать в качестве параметров генерации.

На данный момент создан алгоритм генерации принципом которого является: ФС у элементов в каждом диапазоне должен быть минимально возможным. Выбор данного принципа обусловлен возможностью контролировать соблюдение количества ФС для диапазона.

Созданный алгоритм работает следующим образом. Предварительно создается заданное количество элементов без связей и размещение их по диапазонам. Далее нужно удовлетворить минимум ФС для каждого элемента в соответствии с диапазоном в котором он размещён. Это действие выполняется в два шага: первый шаг останавливается, когда заканчиваются элементы с минимумом ФС то есть, каждый из элементов имеет тот минимум связей, который возможен в диапазоне в котором расположен данный элемент. Второй шаг заканчивается когда отсутствуют элементы с неудовлетворенным минимум ФС. Далее элементы соединяются, пока у диапазонов есть свободные ФС и в схеме возможно создать связь.

**Выводы по главе 2**

Разработан алгоритм формирования исходных данных для генератора коммутационных схем, который позволяет инженеру проектировщику задавать такие параметры коммутационной схемы которые позволяют сгенерировать желаемую схему электронного устройства с заданной элементной базой, а именно компоненты с конструктивными характеристиками и их типовые схемы включения. Представлен принцип формирования входных данных.

## 3 Программный комплекс генерации коммутационных схем

Функционирование генератора коммутационных схем должна быть обеспечено работай таких вспомогательных программных модулей как:

1) модуль работы с файлами;
2) анализатор коммутационных схем;
3) последовательно-итерационный алгоритм размещения;
4) модуль визуализации результата размещения.

На рисунке 3.1 представлена схема  движение информации между программами.

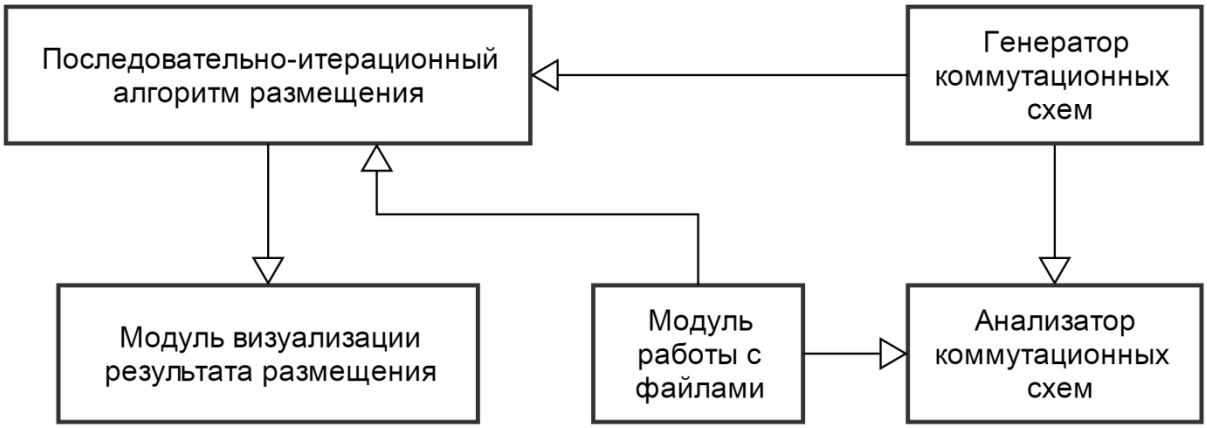



Рисунок 3.1 – Схема  взаимодействия вспомогательных программ и генератора комутационных схем

Модуль работы с файлами осуществляет чтение коммутационных схем созданных в профессиональной среде САПР.

Анализатор коммутационных схем выполняет роль поставщика параметров для генерации, тестирование схем созданных генератором и профессиональной средой САПР.

На базе Реализованного последовательно-итерационного алгоритм размещения выполняется тестирование созданных коммутационных схем.

Модуль визуализации результата размещения отображает результат размещения. Может работать с различными алгоритмами размещения. Используется при тестировании созданных коммутационных схем.

Генератор коммутационных схем - основной программный модуль.

Программный комплекс реализован на языке программирования C# в среде разработке Visual Studio с использованием стандартных средств разработки оконных приложений WPF.

### 3.1 Модуль работы с файлами

Модуль работает с файлами содержащие набор узлов, в каждом из которых перечислены элементы имеющие связь между собой. Данные файлы представляют собой файлы списка соединений, описывающие в текстовом виде граф коммутационной схемы (net-лист). Файлы могут быть сформированы в любой современной САПР печатных плат.

Диаграмма классов модуля представлена на рисунке 3.2.

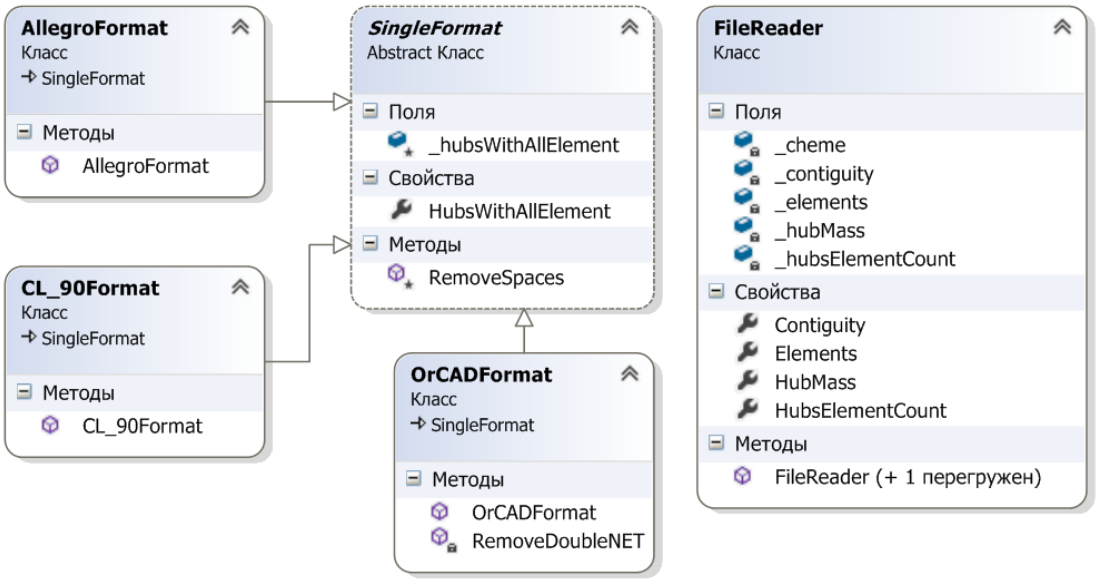



Рисунок 3.2 – Диаграмма классов модуля работы с файлами

Для работы с модулем используется класс FileReader, в конструктор которого передаётся путь к файлу. Используя свойство *Elements* можно получить список всех элементов и основываясь на длине списка их количество. С помощью *HubMass* можно увидеть содержание узлов без повторяющихся элементов, *Contiguity* предоставляет матрицу смежности элементов, это главный продукт работы данного модуля.

Класс SingleFormat объединяет в себе общие способы обработки информации файлов коммутационных схем.

### 3.2 Программная реализация алгоритма размещения

Все математические преобразования над схемой КП вынесены в отдельный класс *CoherentIterative*, который представлен на рисунке 3.3. Для работы класса требуется коммутационная схема, которую предоставляет модуль работы с файлами.

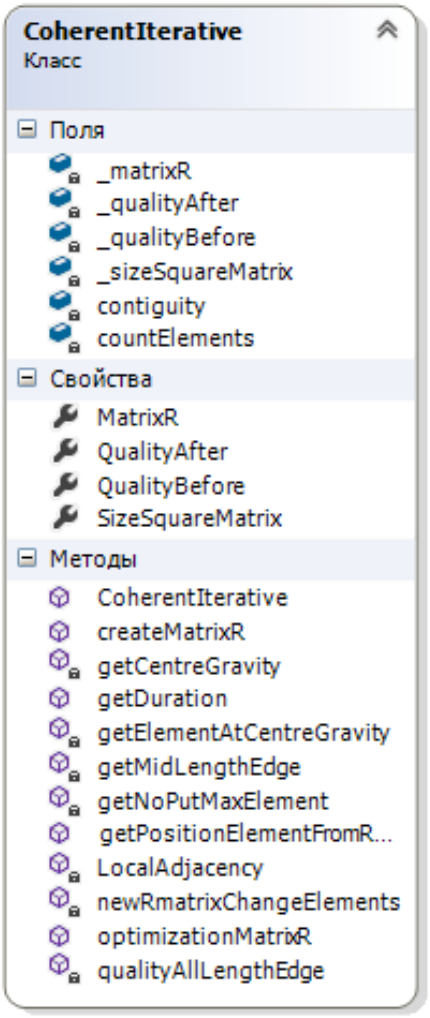



Рисунок 3.2 – Класс реализующий последовательно-итерационный алгоритм размещения.

Конструктор класса имеет обязательный параметр которым является матрица смежности, при инициализации класса создаётся пустая матрица КП, которая имеет всегда квадратную форму, если элементов меньше чем размер матрицы, то оставшиеся ячейки будут пустыми.

Свойство *SizeSquareMatrix* это размер матрицы КП, это значение вычисляется при инициализации класса.

Свойства *QualityBefore* и *QualityAfter* это показатель качества схемы КП до итерационной части алгоритма и после соответственно.

Свойство *MatrixR* это матрица КП, продукт работы данного класса. Данное свойство изменяется в двух методах, первый из них это *createMatrixR,* выполняет последовательный этап алгоритма, и *optimizationMatrixR* выполняет итерационный этап алгоритма.

Метод *LocalAdjacency* в качестве обязательного параметра принимает номер элемента. Позволяет узнать с какими элементами он смежен и количество связей между ними.

Метод *getDuration* в качестве обязательного параметра принимает номера элемента двух элементов между которыми требуется вычислить расстояние, порядок элементов не важен, и в качестве не обязательного параметра принимает матрицу КП, если не задавать этот параметр то по умолчанию берётся текущая матрица. Метод производит вычисление по формуле $d_{ij} = \left|X_i - X_j\right| + |Y_i - Y_j|$, где i, j это номера элементов.

Метод *getPositionElementFromRmatrix* в качестве обязательного параметра принимает номер элемента и в качестве не обязательного параметра принимает матрицу КП, если не задавать этот параметр то по умолчанию берётся текущая матрица. Метод вычисляет позицию элемента в матрице КП.

Метод *getMidLengthEdge* в качестве обязательного параметра принимает номер элемента и в качестве не обязательного параметра принимает матрицу КП, если не задавать этот параметр то по умолчанию берётся текущая матрица. Метод вычисляет среднею длину ребра элемента.

Метод *qualityAllLengthEdge* в качестве не обязательного параметра принимает матрицу КП, если не задавать этот параметр то по умолчанию берётся текущая матрица. Метод вычисляет качество матрицы КП по критерию общей длины рёбер, меньшее значение предпочтительно.

Метод *getCentreGravity* в качестве обязательного параметра принимает номер элемента и в качестве не обязательного параметра принимает матрицу КП, если не задавать этот параметр то по умолчанию берётся текущая матрица. Метод находит координаты центра тяжести элемента и округляет их до значений конкретной ячейки.

Метод *getElementAtCentreGravity* в качестве обязательного параметра принимает координаты центра тяжести и в качестве не обязательного параметра принимает матрицу КП, если не задавать этот параметр то по умолчанию берётся текущая матрица. Метод находит все элементы находящиеся от центра тяжести на расстоянии не более одной ячейки.

Метод *newRmatrixChangeElements* в качестве обязательного параметра принимает номера элемента двух элементов. Возвращает новую матрицу КП в которой два элемента поменялись местами.

Метод *getNoPutMaxElement* на основании текущего состояния матрицы КП возвращает номер элемента удовлетворяющий критерию из последовательной части алгоритма.

Метод *createMatrixR реализут* первый этап работы алгоритма, продуктом работы метода является матрица в которой первый элемент выбран случайным образом, а остальные в соответствии с последовательной часть алгоритма.

Метод *optimizationMatrixR* реализует итерационную часть работы алгоритма, продуктом работы метода является матрица КП оценка качества которой должна улучшится.

### 3.3 Модуль визуализации результата размещения

Для создания модуля используется стандартный набор элементов из C# WPF. Для работы модуля создано два класса, которые представляют из себя окна с элементами интерфейса. Классы представлены на рисунке 3.2.

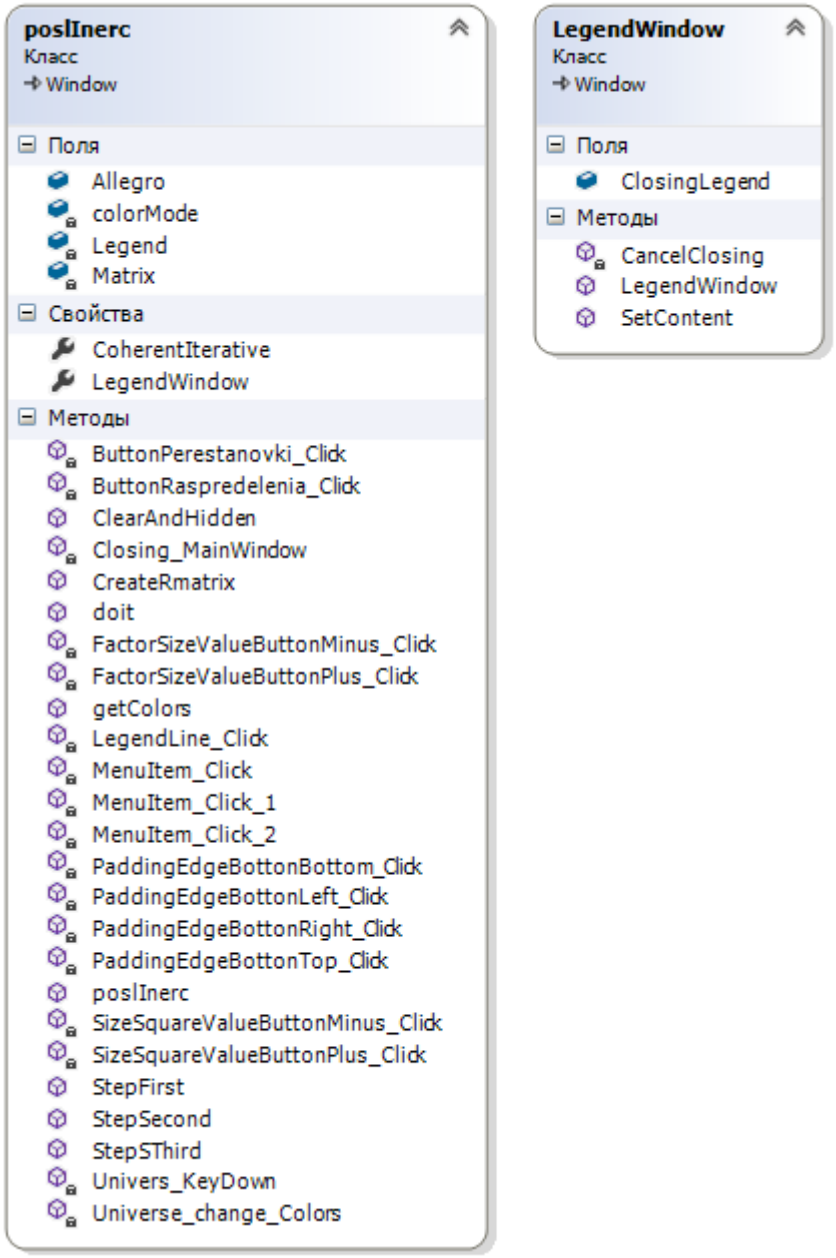



Рисунок 3.3 – диаграмма классов модуля визуализации результата размещения

Модуль визуализации использует модуль реализации алгоритма размещения и модуль работы с файлам, схема использования представлена на рисунке 3.4.

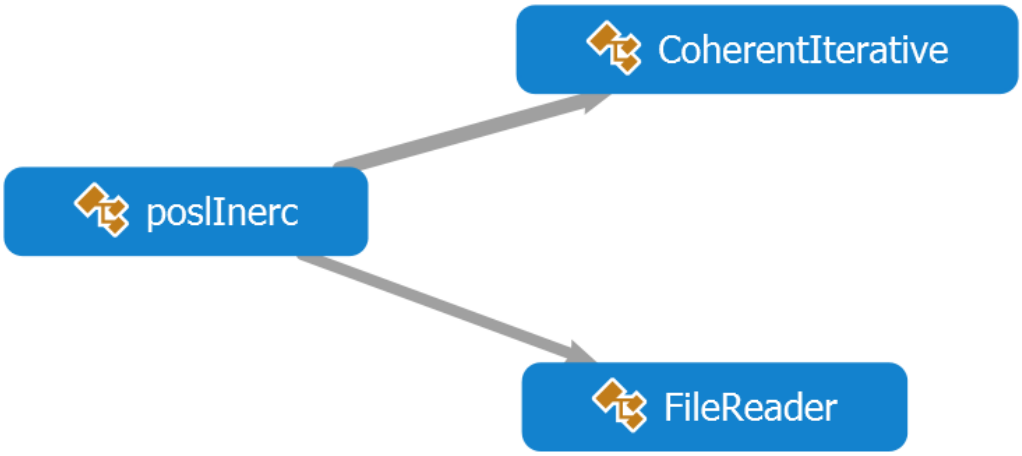



Рисунок 3.4 – Схема взаимодействия модулей

В процессе работы программы результат модуля работы с файлами загружается в реализованный алгоритм размещения

### 3.3.1 Графический интерфейс

Графический интерфейс реализован с целью предоставить пользователю возможность увидеть результат работы класса из предыдущей главы. Благодаря интерфейсу пользователю доступны возможности касающиеся изменения характера отображения схемы КП. На рисунке 3.5 изображён результат работы программы.

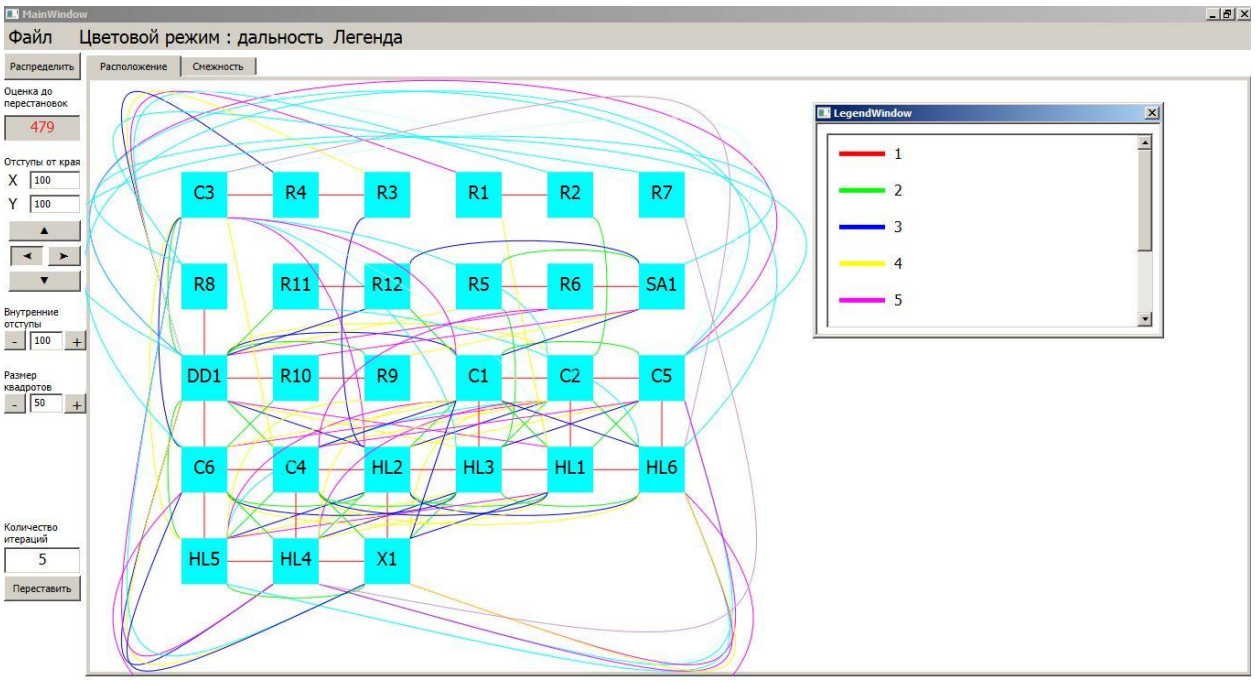



Рисунок 3.4 – Интерфейс программы

Для изменения характера отображения схемы КП имеются следующие возможности: изменение отступов от верхнего и левого края отображения схемы КП, изменение отступов между элементами схемы КП, размер элементов. Благодаря функции "Цветовой режим" в сочетании с окном отображающим легенду линий, пользователь может получить информацию о количестве связей между элементами и расстояние между элементами.

### 3.3.2 Пункт меню: файл

Данный пункт меню введён в программу для того чтобы пользователь работал имел возможность открывать файлы для работы , очищать рабочую область и завершать работу программы. На рисунке 3.5 показан пункт в развёрнутом виде.

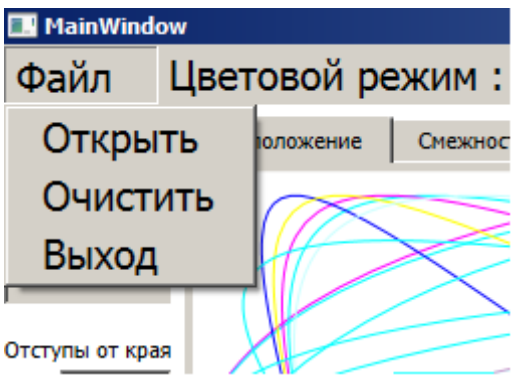


Рисунок 3.5 – Пункт меню "Файл" и его подпункты

При нажатии на "Открыть" появляется окно с выбор файла для дальнейшей работы. Подпункт "Очистить" скрывает многие элементы интерфейс и привести характеристики визуализации к начальным. "Выход" завершает работу программы.

### 3.3.3 Пункты меню: Цветовой режим и Легенда

Изменяя цветовой режим пользователь получает различную информацию и текущем состоянии схемы КП. На рисунке 3.6 показан пункт меню "Цветовой режим". Далее будут рассмотрены все подпункты данного пункта меню и их взаимодействие с окном легенды линий.

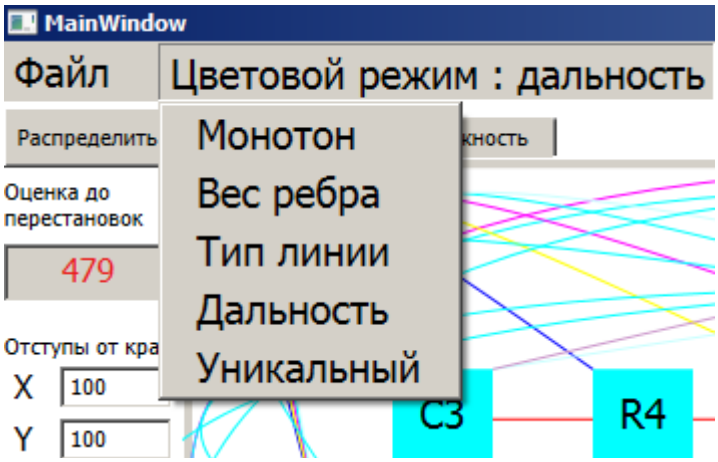



Рисунок 3.6 – Пункт меню "Цветовой режим" и его подпункты

"Монотон" отображает наличие связи между элементами, в окне легенды присутствует соответствующий комментарий.

"Вес ребра" отображает количество связей между элементами, в окне легенды присутствует количество записей равное максимальному числу связи между элементами и числовой показатель напротив цвета линий.

"Тип линии" отражает способ построения линии, в окне легенды присутствует название типа линии напротив каждого цвета. Благодаря данному режиму лучше видны элементы лежащие находящиеся на строго вертикальной

или горизонтальной прямой линии. это режим был добавлен благодаря различному характеру в построении линий.

"Дальность" отображает на сколько удалены элементы от друг друга, в окне легенды присутствует целочисленная характеристика расстояния напротив каждого цвета. Благодаря данной возможности пользователь видит на сколько элементы удаленный друг от друга.

"Уникальный" отображает различные цвета линий, в окне легенды присутствует соответствующий комментарий. Данная возможность добавлена с целью максимально контраста цветового разброса линий.

### 3.3.4 Отступ от края

Данная возможность позволяет пользователю перемещать схему КП в случае если она не помещается в окне программы. На рисунке 3.7 изображён данный элемент интерфейса.

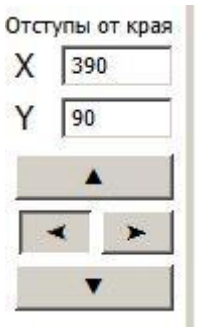



Рисунок 3.7 – Управление позицией схемы КП

Чтобы передвинуть схему КП используйте стрелку с соответствии нужным направлением, или же вводите нужные числа вручную и используйте клавишу "Enter" для применения параметров отступа.

### 3.3.5 Внутренние отступы

При увеличении внутренних отступов увеличивается расстояние между элементами, но изменяется лишь визуальное отображение и не как не влияет на результат работы алгоритма размещения. Данная возможность может помочь

пользователю в случаем детального рассмотрения схемы КП. На рисунке 3.8 показан результат отображение с увеличенным внутренним отступом.

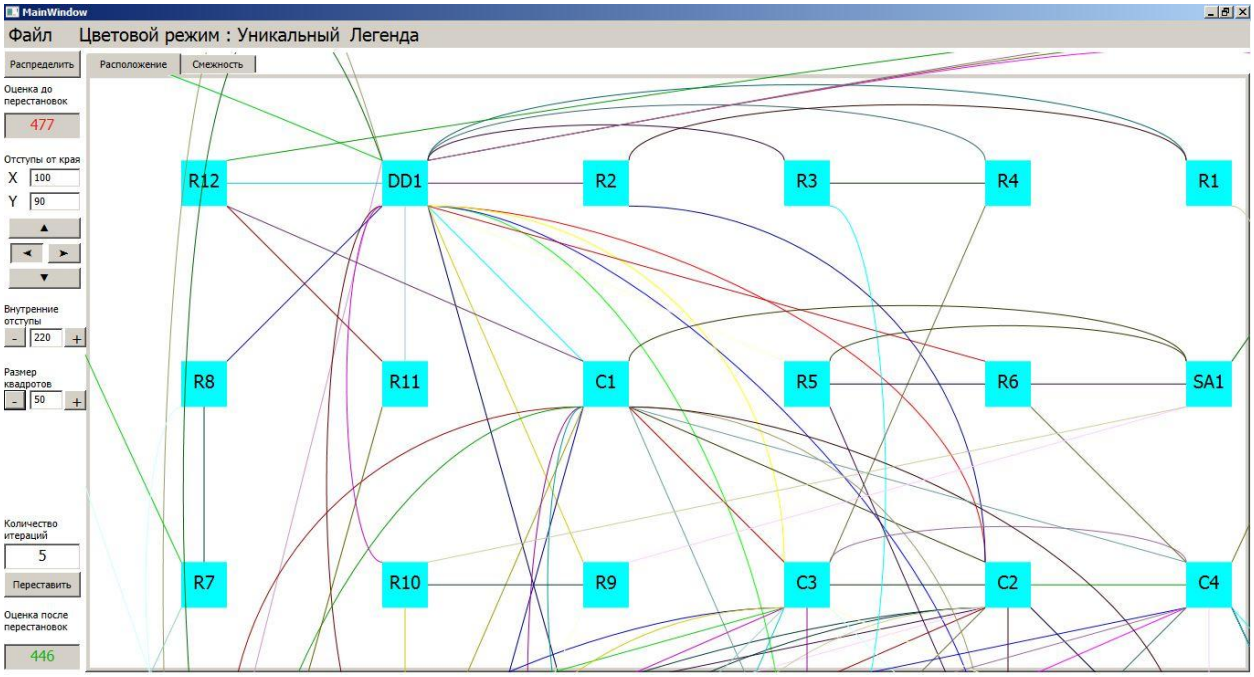



Рисунок 3.8 – Окно программы с увеличенным расстоянием между элементами

### 3.3.6 Размер элементов

Изменение размера элементов может пригодится когда требуется визуально сжать схему КП или же для более детального рассмотрения. Элемент интерфейса изменяющий размер элементов изображён на рисунке 3.9.

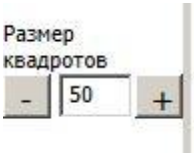



Рисунок 3.9 – Элемент интерфейса изменяющий размер элементов

Что изменить размер элементов используйте кнопки с соответствующим символом или же вводите нужные числа вручную и используйте клавишу "Enter" для применения параметра размера элементов. На рисунке 3.10 изображен пример изменения этой визуальной характеристики.

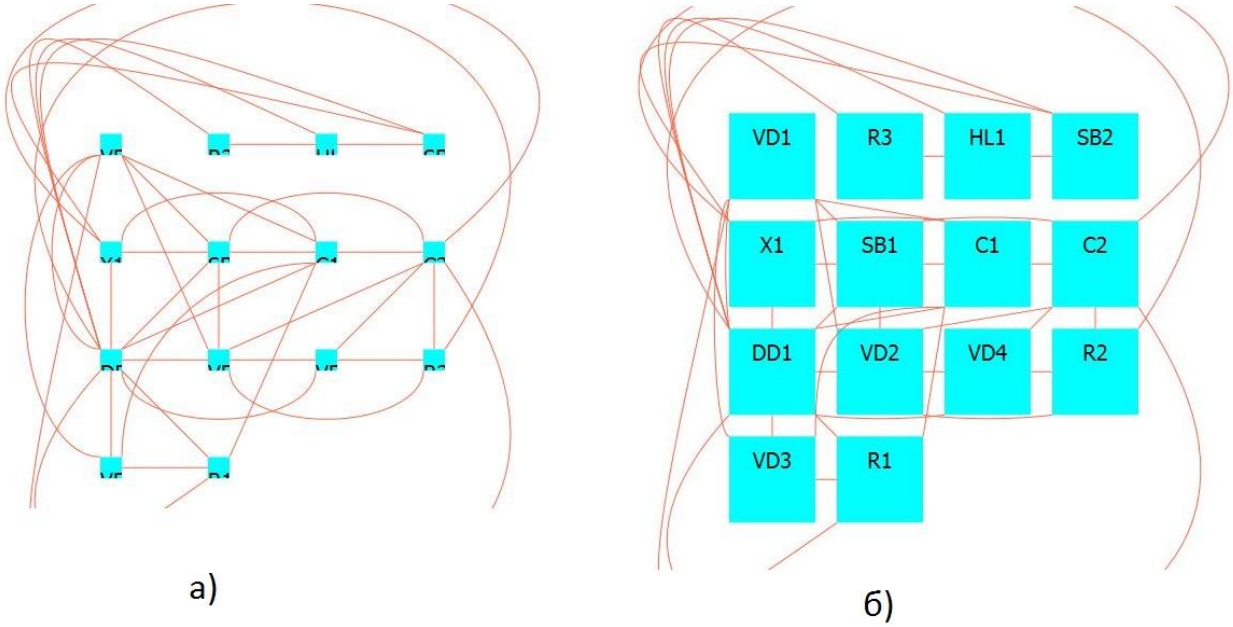



Рисунок 3.10 – Пример изменения размера элементов, а) элементы уменьшены, б) элементы увеличены

### 3.3.7 Отрисовка схемы КП. Оценка качества

Для отрисовки схемы КП используются две кнопки "Распределить" и "Переставить", данные кнопки выполняют последовательный и итерационный этапы алгоритма соответственно. На рисунке 3.11 показаны кнопки и оценки качества.

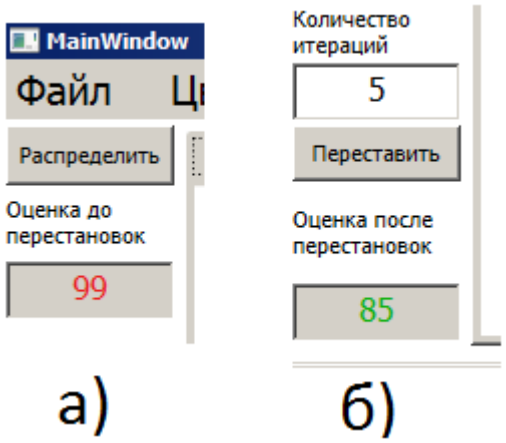



Рисунок 3.11 – Кнопки и оценки качества, а) последовательная часть алгоритма и оценка до итерационной части алгоритма , б) итерационная часть алгоритма и оценка после её выполнения

### 3.4 Анализатор коммутационных схем

В анализаторе коммутационных схем собраны различные параметры, которые позволяют подробно описать коммутационную схему. На рисунке 3.12 представлена диаграмма классов анализатора.

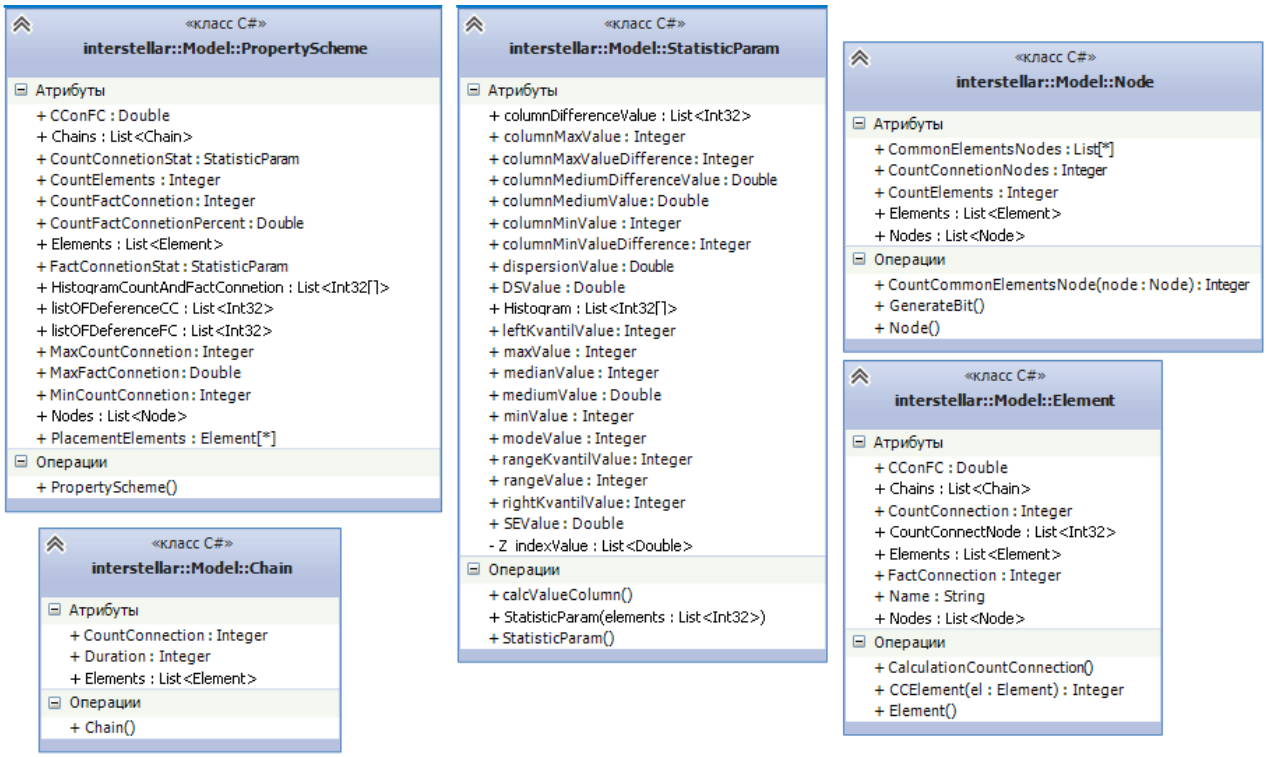



Рисунок 3.12 – Диаграмма классов

Класс StatisticParam используется для полного анализа ФС и КС в схеме, реализует различные статистические характеристики, такие как:

1) максимальное значение выборки;
2) размах выборки;
3) средние значение в выборке;
4) мода выборки;
5) медиана выборки;
6) дисперсия выборки;
7) среднеквадратическое отклонение;
8) квантили выборки;
9) стандартная ошибка среднего.

В класс PropertyScheme собраны все выявленные характеристики схемы и полный перечень всех элементов, узлов, связей.

### 3.5 Генератор коммутационных схем

Компоненты состоволяющие генератор комутационных схем представлены на рисунке 3.13.


Рисунок 3.13 – Схема взаимодействия компонентов

Все вспомогательные программы включены в testFacotCenect. А также данный компонент содержит реализацию графического интерфейса, механизм последовательной установки параметров генерации и механизм уточнения.

Для реализации механизма уточнения создан инструмент, который вынесен в отдельный компонент testFacotCenect.Controls.

Для реализации генератора коммутационных схем потребовалось создать особую структуру данных, которая вынесена в testFacotCenect.Model.

Диаграмма зависимостей основного модуля программы представлена на рисунке 3.14.

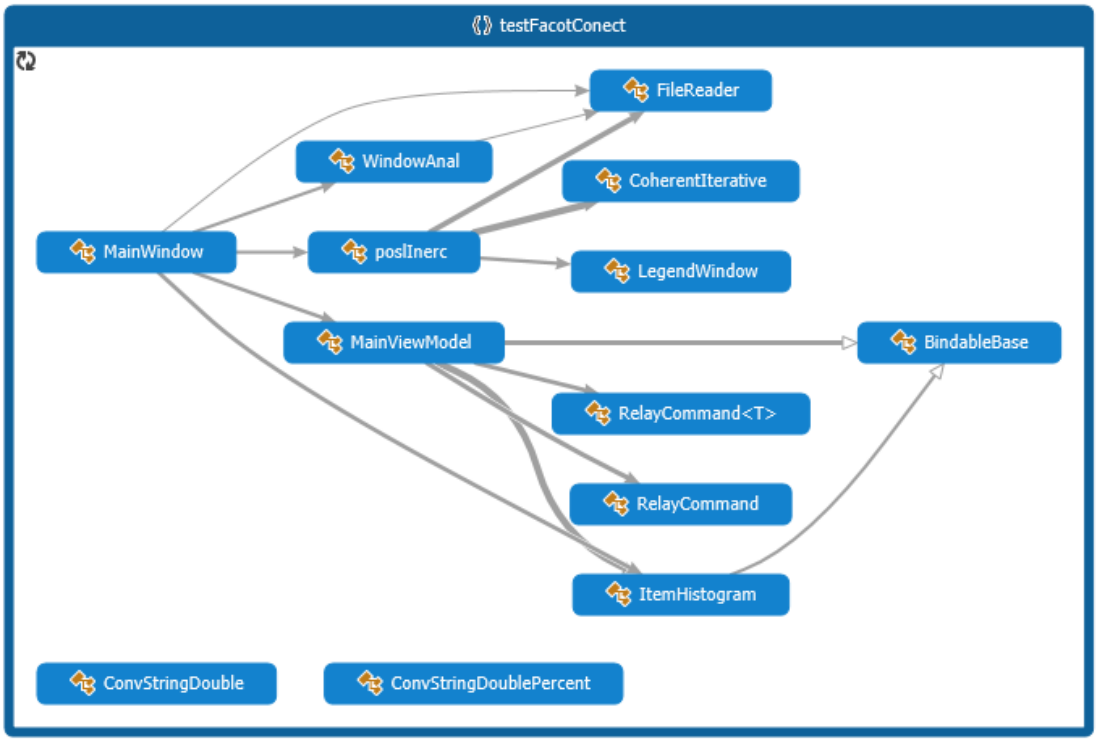



Рисунок 3.14 – Диаграмма зависимостей основного модуля программы

Диаграмма классов основного модуля программы представлена на рисунке 3.15.

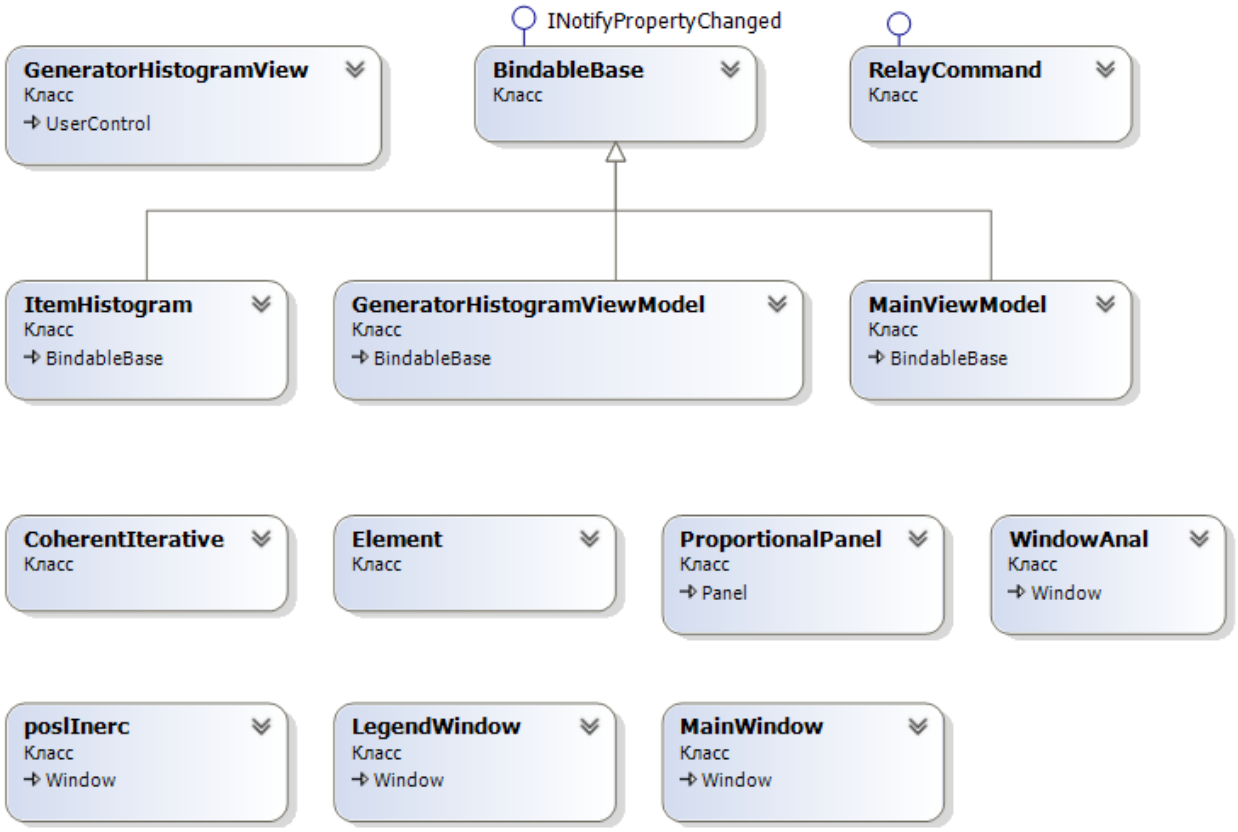



Рисунок 3.15 – диаграмма классов основного модуля программы

Часть программы реализована в соответствии паттерна MVVM, поэтому используется интерфейс INotifyPropertyChanged и ICommand.

### 3.5.1 Структуры данных коммутационной схемы

Самым простым и очевидным способ представления коммутационной схемы в терминах программирования является двумерный массив. Однако, в процессе разработки алгоритма генерации коммутационной схемы выявлено, что данная структура данных имеет низкий уровень (простота алгоритмической реализации) удобства. Принадлежность элементов к конкретным диапазонам является важной частью требуемой структуры данных. Основное требование к структуре данных это возможность проведения анализа схемы либо ее части. Создана иная структура соответствующая принципам ООП, которая позволила снизить сложность разработки и позволила комфортно манипулировать информацией о схеме. На рисунке 3.16 представлена упрощенная диаграмма зависимостей с созданной структуры данных.

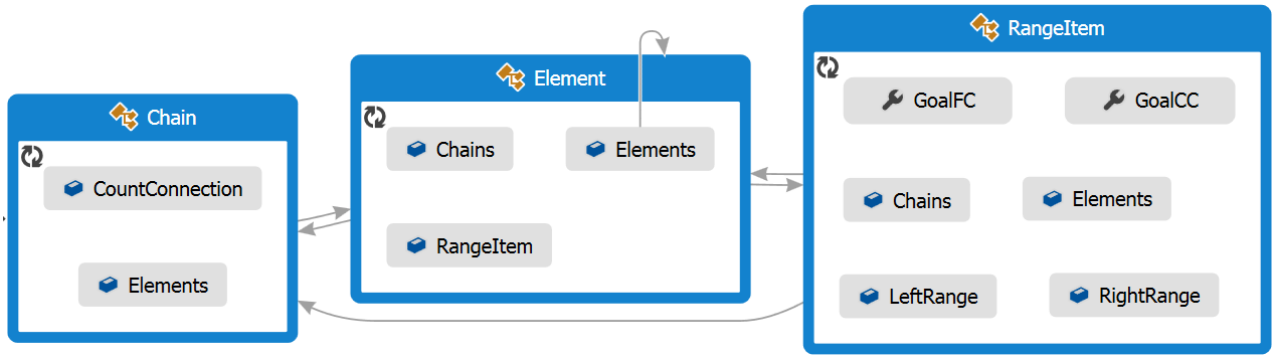



Рисунок 3.16 – Упрощенная диаграмма зависимостей структуры данных для генерации коммутационной схемы

Chain - связь между элементами. Хранит в себе два элемента между которыми существует связь в поле Elements, можно интерпретировать как ФС. Поле CountConnection можно интерпретировать как КС.

Element - элемент коммутационной схемы. Поле Elements содержит список всех элементов, с которыми связан текущий элемент. Поле Chains содержит список всех связей в которых участвует текущий элемент. Поле RangeItem содержит диапазон к которому относится элемент.

RangeItem - диапазон, содержит данные генерации. Поля LeftRange и RightRange левая и правая граница диапазона по ФС соответственно. Поле Chains содержит список всех связей, которые относятся к текущему диапазону. Поле Elements содержит список всех элементов, которые входят в текущий диапозон.

Созданная структура данных является переходной от параметров генерации к коммутационной схеме. Это осуществляется за счет помещения элементов в диапазоны.

На рисунке 3.17 изображена диаграмма классов созданной структуры данных.

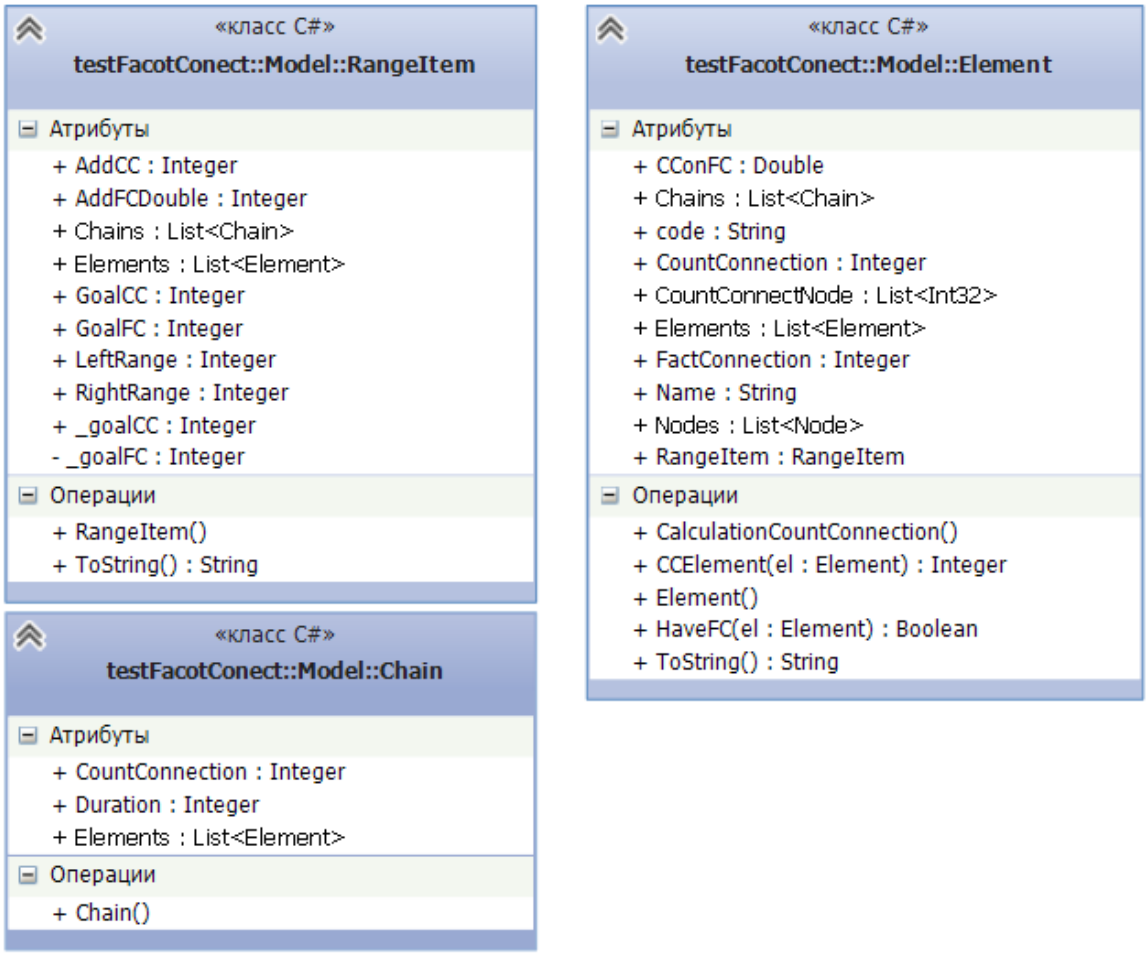



Рисунок 3.17 – Диаграмма классов структуры данных

Преимуществом данной структуры является высокий уровень осведомленности каждого элемента структуры о своём назначении. То есть не нужно производить поиск информации, так как каждый структурный элемент может сообщить свои качественные и количественные характеристики.

Недостаток данной структуры заключается в дублировании информации в момент ее добавления и изменения.

### 3.5.2 Механизм формирования данных генерации

Механизм содержится в двух файлах, приведённых на рисунке 3.15, и представляет из себя интерактивный процесс взаимодействия с пользователем, результатом которого являются сформированные параметры генерации.


Рисунок 3.18 – Классы с элементами механизма формирования данных

На рисунке 3.19 представлена диаграмма активности процесса установки параметров.

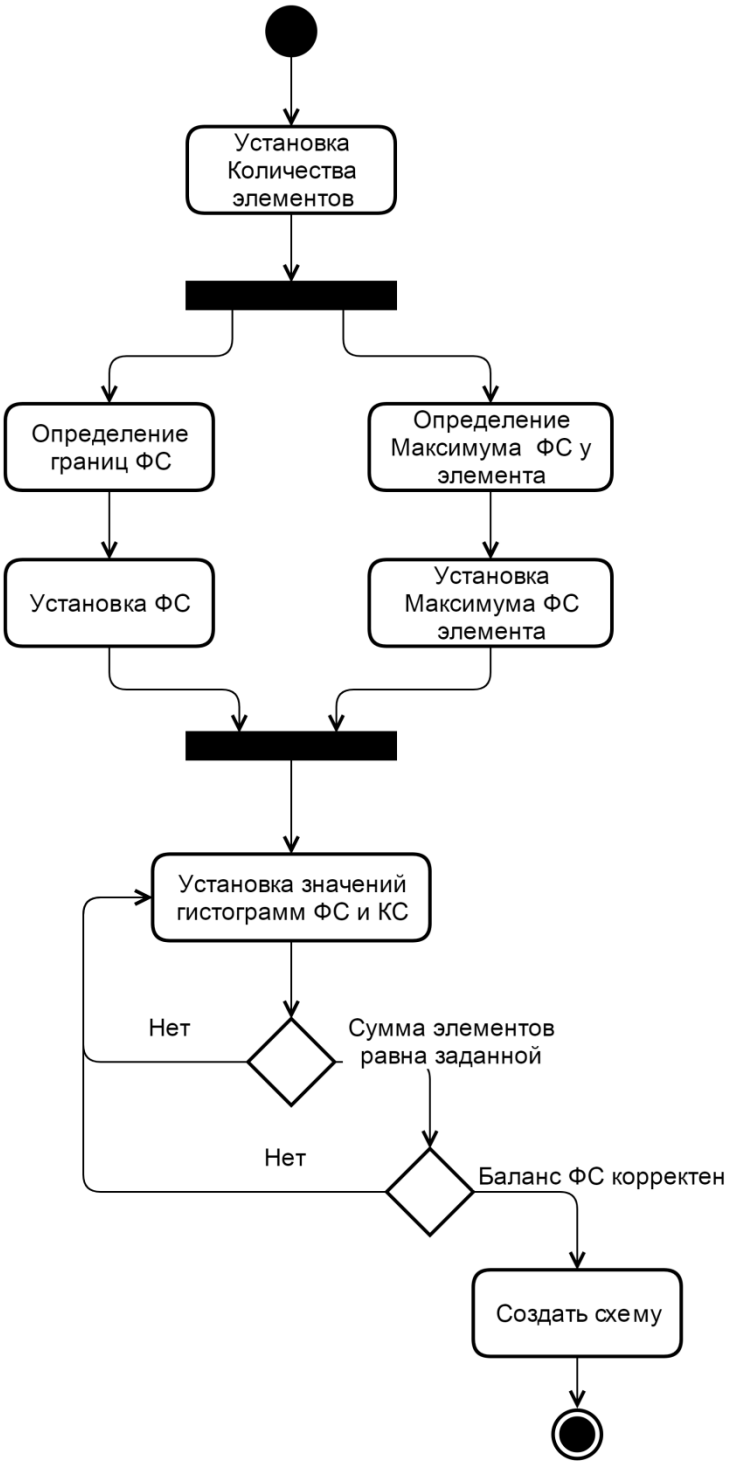



Рисунок 3.19 – Диаграмма активности установки параметров генерации

Параметр "количество элементов" устанавливается первым, так как не имеет ограничений и является определяющим для других элементов.

После установки параметра "количество элементов" пользователь, взаимодействуя с интерфейсом, производит просчет границ параметров при

помощи функции MainWindow. Button_Click_1. Функция реализует формулы (2.1) и (2.2).

Далее устанавливается параметры "ФС" и "Максимум ФС элемента", в соответствии с установленными границами.

Далее создается первоначальная гистограмма. Последующие изменение гистограммы управляется механизмом уточнения, который является частью механизма формированию данных генерации.

После установки гистограмм ФС и КС выполняются проверки на корректность. Первая проверка осуществляется в функции MainViewModel.CheckArightSumCountElementFunc. Назначение функции выявить равенство суммы количества элементов в диапазонах и параметром" количество элементов". Если проверка не пройдена блокируется осуществление следующей проверки.

Вторая проверка нужна для соответствия суммы ФС диапазонов указанной в параметре "ФС". Осуществляется в функции MainViewModel.CheckBalanceFCFunc. Функция реализует формулы (2.3) и (2.4).

### 3.5.3 Механизм уточнения

Механизм используется для работы с гистограммой и представляет из себя процесс описания схемы при помощи деления диапазона на две части.

Механизм очень гибкий. Пользователь самостоятельно решает каким образом он желает взаимодействовать с диапазонами. При каждом разделении диапазона пользователь может перераспределять КЭ, ФС и КС для двух новых диапазонов, а возможно и всей гистограммы. Или же пользователь может сначала создать все диапазоны и потом распределить КЭ, ФС и КС.

Для обеспечения данной возможности создан пользовательский элемент управления, диаграмма зависимости которого представлена на рисунке 3.20.

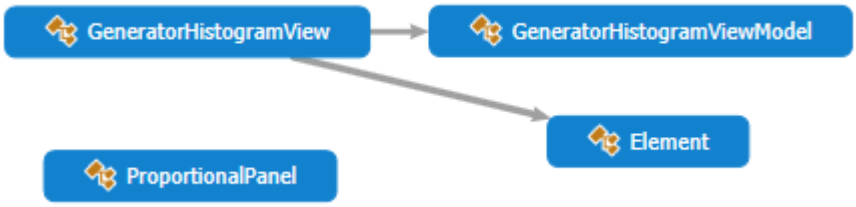



Рисунок 3.20 – Диаграмма зависимости элемента управления

Диаграмма классов элемента управления представлена на рисунке 3.21.


Рисунок 3.21 – Диаграмма классов элемента управления

Создание элемента управления обусловлено отсутствием бесплатных и подходящих аналогов для реализации механизма уточнения.

### 3.5.4 Алгоритм генерации

Алгоритм использует структуру данных из главы 3.4.1. Создание схемы происходит в 4 этапа, 3 из которых формируют связи между элементами, а 4 заполняет созданные связи КС. Диаграмма классов представлена на рисунке 3.22.

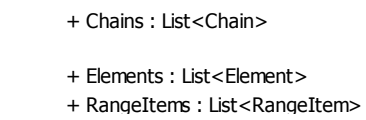



Рисунок 3.22 – Диаграмма классов алгоритма генерации

После передачи параметров генерации используется функция GenerateCheme. Первое что нужно сделать для генерации схемы это создать минимальное количество связей элемента в соответствии с диапазоном, в который он помещён. Это осуществляют функции StepOneNewNEW и StepTwoNewNEW. На основании заложенных принципов в главе 2.4. функция StepThreeNEW осуществляет распределение связей между диапазонами. Для заполнения связей КС используется функция AddCC.

### 3.5.5 Интерфейс

В целях предоставления более комфортного процесса работы пользователя, создан графический интерфейс, который представлен на рисунке 3.23, где зона а - выбор стороны создания нового диапазона; зона б - граница минимума ФС; зона в - граница максимума ФС; зона г - граница максимума ФС элементов; зона д - выполнение проверок корректности данных генерации; зона е - выбор значения разделения диапазона; зона и - параметры диапазона.

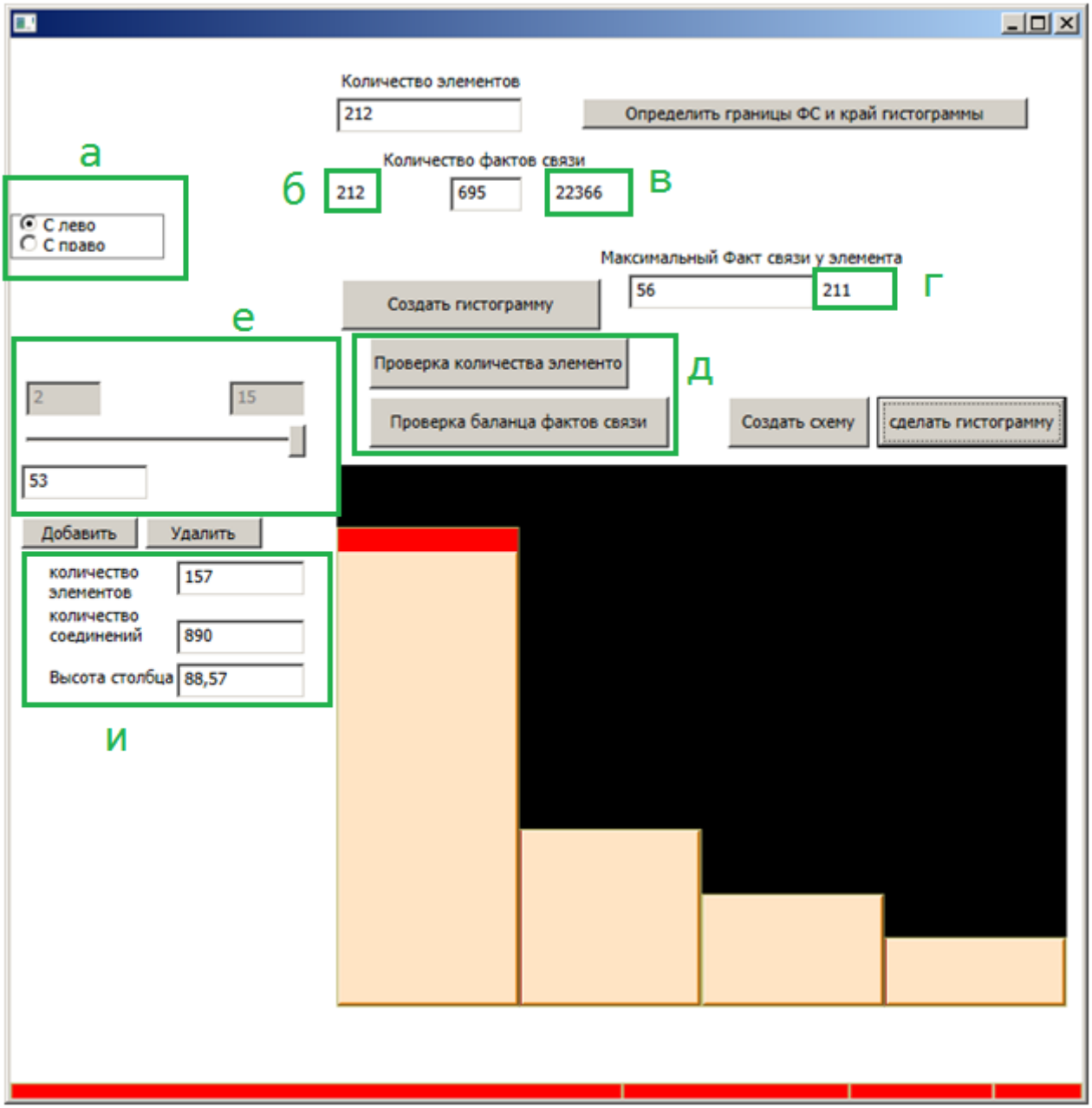



Рисунок 3.23 – Интерфейс генератора коммутационных схем

Последовательность боты с интерфейсом следующая:

1) ввод количества элементов;
2) создание границ кнопкой "Определить границы ФС и край гистограммы";
3) ввод "ФС" и "Максимальный факт связи у элемента";
4) создание начальной гистограммы кнопкой "Создать гистограмму";
5) создание диапазонов гистограммы и установка их параметров.

6) создание схемы кнопкой "создать схему"

Процесс создание нового диапазона имеет следующею последовательность :

1) выбор диапазона для разделения;
2) выбор стороны создания нового диапазона, элемент управления представлен на рисунке 3.23.а;
3) выбор значение разделения выделенного диапазона, представлено на рисунке 3.23.е;
4) создание нового диапазона кнопкой "Добавить";
5) установка значения параметров диапазонов, представлено на рисунке 3.23.и.

Созданный интерфейс реализует функционал создания данных генерации коммутационной схемы. А также, благодаря заложенным механизмам предотвращает установку некорректных данных генерации.

### 3.6 Тестирование

Проведена апробация методики генерации коммутационной схемы, которая показывает совпадения сгенерированной схемы созданной по заданной реальной коммутационной схеме воплощённой на печатной плате приведенной на рисунке 3.24.

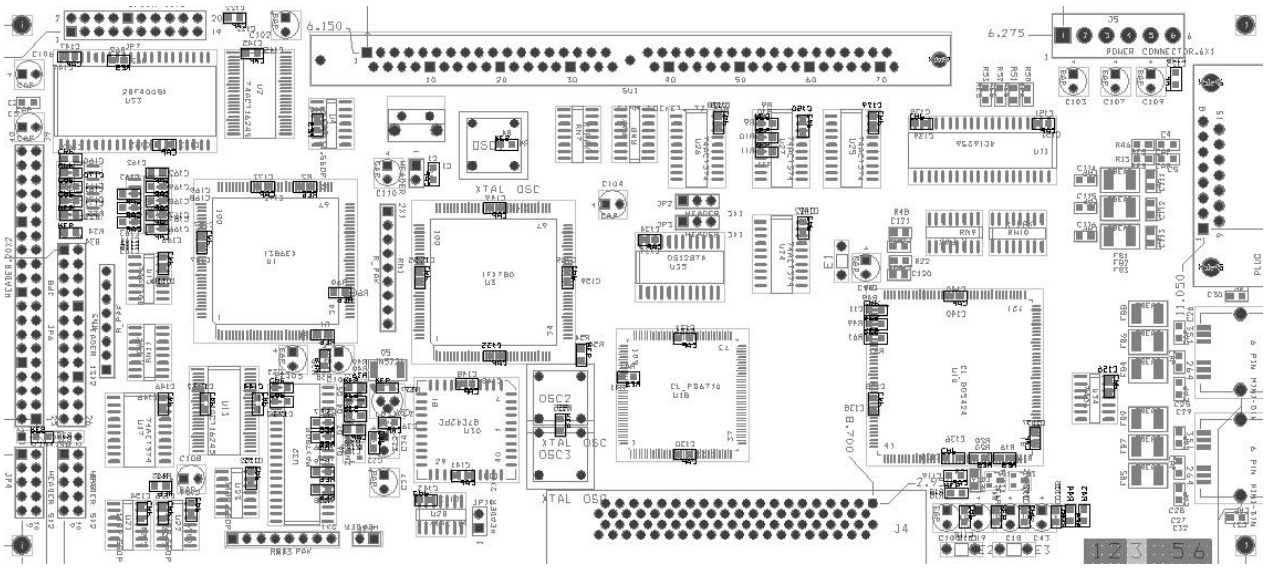


Рисунок 3.24 – печатная плата с реализованной коммутационной схемой

Исходным параметром является количество элементов составляющим 212 штук, все параметры для реальной схемы были просчитаны с использованием программного модуля "анализатор коммутационных схем". Следующим шагом в апробации был ввод значений параметров схемы в генератор коммутационных схем. Приведены параметры генерации:

— Количество элементов - 212;

— ФС - 695;

— Правая граница - ФС 56.

— Параметры диапазонов:

1) КЭ = 157, дополнительный КС = 890, левая граница = 2, правая граница = 15;
2) КЭ = 50, дополнительный КС = 450, левая граница = 16, правая граница = 29;
3) КЭ = 3, дополнительный КС = 150, левая граница = 30, правая граница = 53;
4) КЭ = 2, дополнительный КС = 200, левая граница = 54, правая граница = 56.

В результате создана схема, которая была также загружена в анализатор коммутационных схем. Результаты сведены в таблицу 3.1.

Таблица 3.1 – Параметры коммутационной схемы

| Параметры схемы | Реальная схема | Сгенерированная схема |
|---|---|---|
| КЭ | 212 | 212 |
| ФС | 695 | 691 |
| КС | 1527 | 1487 |
| максимальное ФС | 56 | 56 |
| максимальное КС | 287 | 162 |
| средние ФС | 6,5566037735849054 | 6,4716981132075473 |
| средние КС | 14,40566037735849 | 14,028301886792454 |

Совпадения результатов составляют:

— ФС - 99%

— КС - 97%

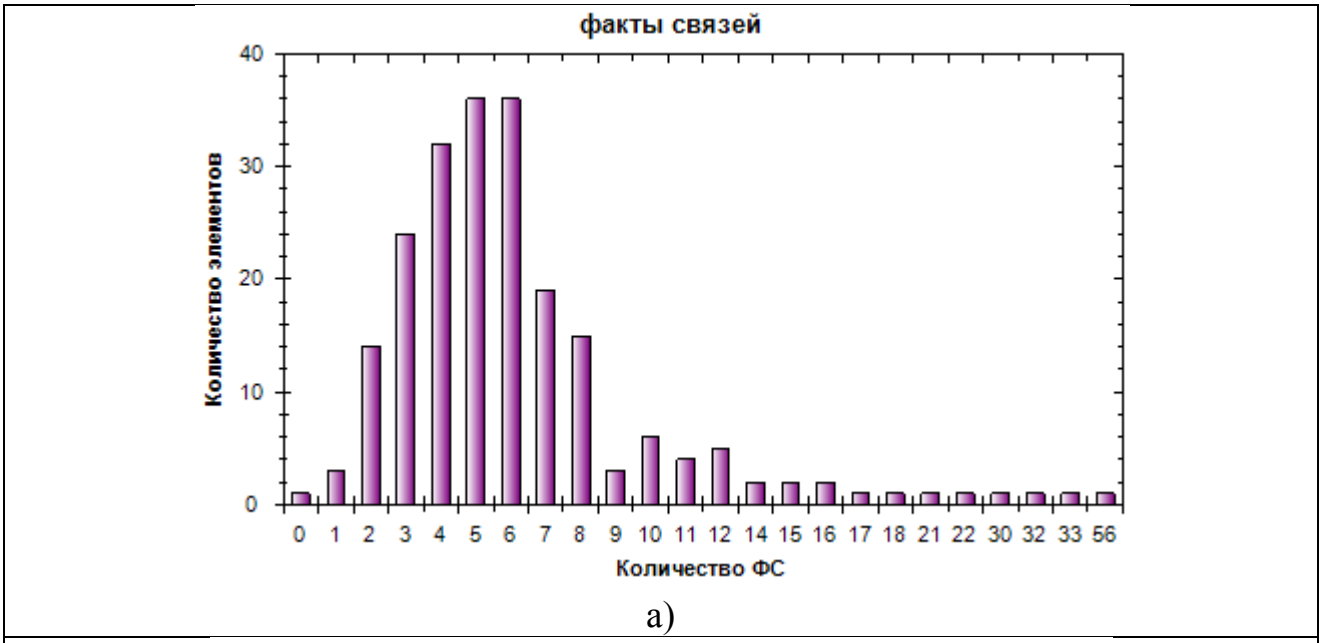
— максимальное ФС - 100%

— максимальное КС - 56

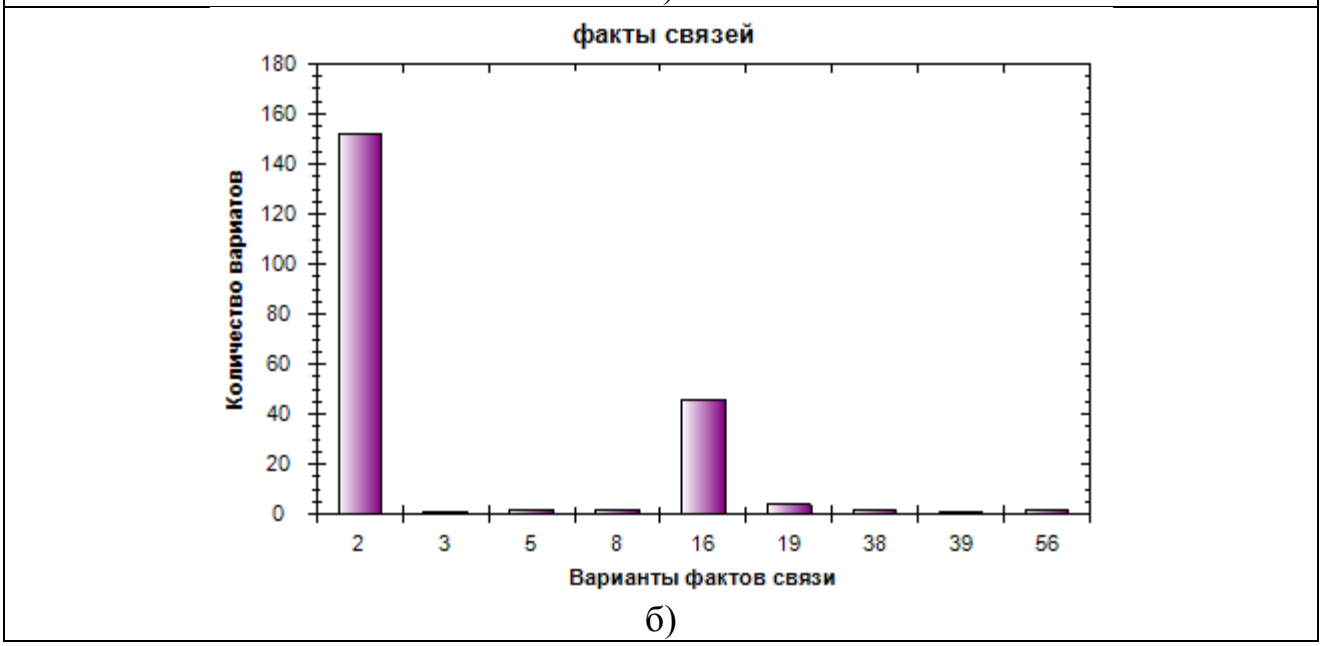
— средние ФС - 98%

— средние КС - 97%

Гистограммы по ФС реальной и сгенерированных схем представлены на рисунке 3.25.




а)




б)

Рисунок 3.25 - Гистограммы ФС. а) реальная схема; б) сгенерированная схема

Гистограммы по КС реальной и сгенерированных схем представлены на рисунке 3.26.

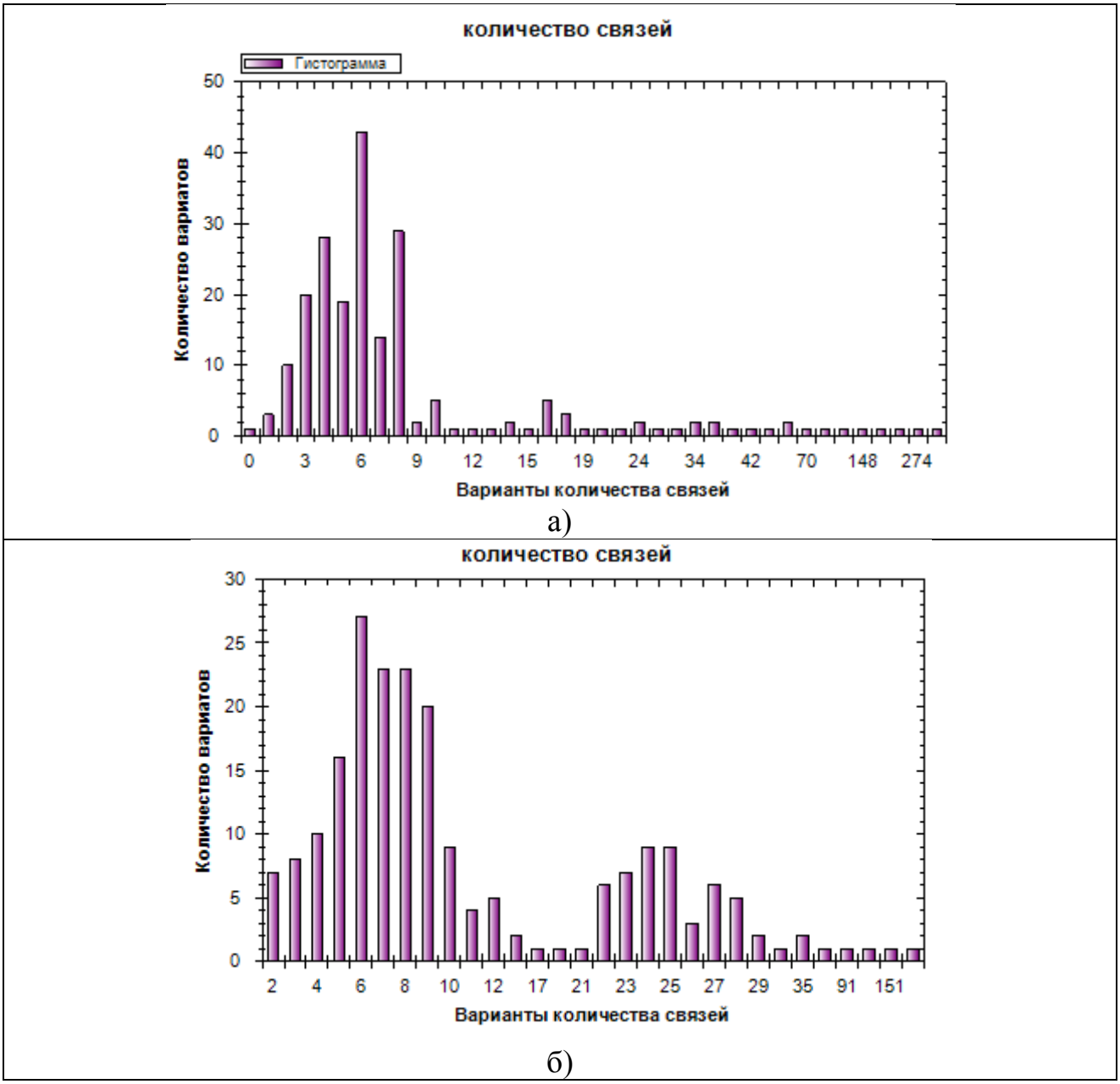



Рисунок 3.26 - Гистограммы КС. а) реальная схема; б) сгенерированная схема

Гистограммы сгенерированной схемы отображают особенность созданного алгоритма генерации, которые обозначены в главе 2.4.

**Вывод по главе 3**

В рамках работы было реализовано несколько вспомогательных программ и модулей, которые обеспечивают работу генератора коммутационных схем:

Анализатор является средством выявлением информации как из реальных коммутационных схем, так и созданных генератором. Для работы с реальными схемами анализатору требуется модуль работы с файлами.

На примере последовательно-итерационного алгоритма размещения осуществлялось тестирование алгоритма на возможность реализации размещения. Визуализация размещения позволила выявить некорректность данных.

Созданы принципы по которым должен работать генератор коммутационных схем:

— минимальный набор данных для генерации схемы;

— постоянный контроль корректности данных для генерации схемы;

— принцип уточнения.

Созданы механизмы осуществляющие соблюдение созданных принципов:

— интерпретация данных для генерации (алгоритм генерации);

— последовательная установка параметров генерации;

— ограничение доступных значений параметра генерации на основании значений ранее установленных параметров.

— контроль подробности параметров генерации.

Реализованы следующие составные части программы:

— модель структуры данных коммутационной схемы;

— алгоритм генерации;

— алгоритм последовательной установки параметров;

— элемент управления интерфейса.

# ЗАКЛЮЧЕНИЕ

Обоснование актуальности исследования было сформулировано на основе необходимости для инженера-конструктора иметь в своём распоряжении большое количество коммутационных схем при проектировании устройств заданного класса с целью обоснования выбора эффективного алгоритма конструирования, в частности, алгоритма размещения.

Разработаны механизмы формирования коммутационных схем на базе математического аппарата формирования данных для генерации, что потребовало введение новых параметров для описания характеристик коммутационных схем, которые позволяют однозначно охарактеризовать связи каждого элемента с каждым внутри коммутационной схемы.

Разработан математический аппарат процесса создания коммутационных схем: введен термин "факт связи", создан алгоритм формирования данных для генерации, создана модель структуры данных для алгоритма генерации, создан алгоритм генерации.

Реализована программа, предоставляющая пользователю возможность генерировать схемы, в которую входит: интерактивный интерфейс предоставляющий пользователю возможность задавать значения параметров генерации, гарантирующий корректность данных, математическое ядро формирования ограничений параметров генерации и генерации коммутационных схем для обеспечения работы генератора коммутационных схем были разработаны вспомогательные программные модули: модуль работы с файлами, анализатор коммутационных схем, последовательно-итерационный алгоритм размещения, модуль визуализации результата размещения.

Апробация разработанного генератора коммутационных схем показала правдоподобность идеи генерации коммутационных схем приближенных к реальным.

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ


1 Алгоритмы и анализ / Томас Х. Кормен [и др.], 3-е изд. : Пер. с англ. − Москва. : ООО "И.Д. Вильямс", 2013. − 1328 с. : ил. − Парал. тит. англ. ISBN 978−5−8459−1794−2 (рус.)

2 Курейчик, В.М. Математическое обеспечение конструкторского и технологического проектирования с применением САПР: учебник для высших учебных заведений / В.М. Курейчик. В.М. − Москва.: Радио и связь, 1990. −352 с.

3 Селютин, В.А. Машинное конструирование электронных устройств: научное издание / В.А. Селютин. - Москва: Советское радио, 1977. - 383 с.

4 Линейка продуктов Cadence OrCAD для проектирования печатных плат» [электронный ресурс]. – Режим доступа: https://www.orcad.com

5 Программное обеспечение Mentor [электронный ресурс]. – Режим доступа: https://www.mentor.com

6 Инструмент разработки электронных устройств Delta Design [электронный ресурс]. – Режим доступа: http://dd.ru

7 Программное обеспечение Altium [электронный ресурс]. – Режим доступа: http://www.altium.com

8 Щеглов, С.Н. Проведение вычислительных экспериментов при принятии решений для графовых моделей в САПР / С.Н. Щеглов // Известия ЮФУ. Технические науки. – 2014. – № 7 (156). – С. 101-108.

9 Курейчик, В.В. Интегрированный алгоритм размещения фрагментов СБИС/ В.В. Курейчик, Вл.Вл. Курейчик // Известия ЮФУ. Технические науки. – 2014. – № 7 (156). – С. 84-93.

10 Запорожец, Д.Ю. Об одном способе кодирования решения для задачи размещения / Д.Ю. Запорожец, Д.В. Заруба, А.А. Лежебоков // Известия ЮФУ. Технические науки. – 2012. – № 11 (136). – С. 183-188.

11 Курейчик, В.В. Иерархический подход при размещении компонентов СБИС/ В.В. Курейчик, Д.В. Заруба, Д.Ю. Запорожец // Известия ЮФУ. Технические науки. – 2014. – № 7 (156). – С. 75-84.

12 Щеглов, С.Н. Использование различных видов моделей эволюции для построения информационных технологий поддержки принятия решений в проектировании / С.Н. Щеглов // Известия ЮФУ. Технические науки. – 2013. – № 7 (144). – С. 47-52.

13 Курейчик, В.В. Биоинспирированный алгоритм разбиения схем при проектировании СБИС / В.В. Курейчик, Вл.Вл. Курейчик // Известия ЮФУ. Технические науки. – 2013. – № 7 (144). – С. 23-29.

14 Лисяк, М.В. Лежебоков, А.А. Алгоритм многокритериального размещения элементов СБИС / М.В. Лисяк, А.А. Лежебоков // Известия ЮФУ. Технические науки. – 2013. – № 7 (144). – С. 70-75.

15 Кулиев, Э.В. О гибридном алгоритме размещения компонентов СБИС / Э.В. Кулиев, А.А. Лежебоков // Известия ЮФУ. Технические науки. – 2012. – № 11 (136). – С. 188-192.

16 Бакало, М.А. К вопросу построения модифицированного алгоритма размещения / М.А. Бакало, В.В Курейчик // Технологический институт Федерального государственного образовательного учреждения высшего профессионального образования "Южный федеральный университет" в г. Таганроге. – 2006. – № 4. – С. 58-71.

17 Григорьев, В.А. Применение системы имитационного моделирования для сравнительного анализа эффективности решения задачи компоновки различными алгоритмами / В.А. Григорьев, В.В. Лебедев, К.А. Карельская // Автономная некоммерческая научно-образовательная организация «Приволжский Дом знаний». – 2016. – № 4. – С. 15-19.

18 СТО 4.2 07 2014. Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Взамен СТО 4.2 07 2012; дата введ. 30.12.2013. – Красноярск, 2013. – 60с.